VISUAL DENSE THREE-DIMENSIONAL MOTION ESTIMATION IN THE WILD

A Dissertation

By

Zhaoyang Lv

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of College of Interactive Computing

Georgia Institute of Technology

December 2019

VISUAL DENSE THREE-DIMENSIONAL MOTION ESTIMATION IN THE WILD

Approved by:

Dr. James M. Rehg, Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Frank Dellaert
School of Interactive Computing
Georgia Institute of Technology

Dr. James Hays
School of Interactive Computing
Georgia Institute of Technology

Dr. Zsolt Kira
School of Interactive Computing
Georgia Institute of Technology,
Georgia Tech Research Institute

Dr. Andreas Geiger
Autonomous Vision Group
Max Planck Institute,
University of Tübingen

Date Approved: July 19, 2019

To everyone who creates the dots and connects the dots

ACKNOWLEDGEMENTS

Since I joined Georgia Tech in the summer of 2014, I have been fortunate to work with many great people in the past five years. Being able to work with great people is the always most rewarding experience in my life. Their excellent work and remarkable personalities have inspired me on what I should improve in myself. I must thank many of them who gave me the advice that impacts my thoughts in research as well as my life.

First, I must thank both of my advisors, Jim Rehg and Frank Dellaert. Frank brought me to Georiga Tech and guided to start in cutting-edge research. Jim encourages to explore the unknowns in the research problems and valuable feedback, which have finally shaped into this thesis. I am very grateful to both Jim and Frank for teaching me how to ask the right questions, how to think deeply in the solutions, how to execute timely, how to communicate with people.

In the last five years, I have also spent some wonderful time in the internships at Nvidia and Max-Planck Institute Intelligent System. During those times, I have been very fortunate to work with great researchers, including Andreas Geiger, Kihwan Kim, Alejandro Troccoli, Deqing Sun, and Jan Kautz. We collaborated on several projects which produces fruitful results. These work turn out to be the essential parts of this thesis. I have also gained a deeper understanding of research projects management, paper writing, and communications. I was able to meet with great researchers, attend seminars, eventually become friends with many of the fellows who share the same passion as me in our work.

I am very thankful to my friends and mentors at Georgia Tech. Zsolt Kira, Chris Beall, Pablo Alcantarilla, and Fuxin Li collaborated with me closely on my very first research project. I also actively discuss research with my lab fellows, including Jing Dong, Yin Li, Abhijit Kundu. These discussions inspired to think more in-depth in my research. I am also grateful to be able to work with James Hays and Dhruv Batra as a teaching assistant in their classes.

At last, I want to thank my parents and girlfriend. They encouraged me to pursue my research dream and always believe in me far more than I do. It is their support that makes my faith to overcome all the difficulties throughout the past five years of research and life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

One of the most fundamental abilities of the human perception system is to seamlessly sense the changing 3D worlds from our ego-centric visual observations. Driven by the modern applications of robotics, autonomous driving, and mixed reality, machine perception requires a precise dense representation of 3D motion with low latency. In this thesis, we focus on the task of estimating absolute 3D motions in the world coordinate in unconstrained environments observed from ego-centric visual information only. The goal is to achieve a fast algorithm that can produce an accurate representation of the densely rich 3D motions.

To achieve this goal, I propose to investigate the problem from four perspectives with the following contributions.

1) Present a fast and accurate continuous optimization approach that solves the scene motions as fixed-a-priori planar segments.

2) Present a learning-based approach that recovers the dense scene flow from ego-centric motion and optical flow, decomposed by a novel data-driven rigidity prediction.

3) Present a modern synthesis of the classic inverse compositional method for 3D rigid motion estimation using dense image alignment.

4) Present a two-view monocular scene flow approach that recovers depth, camera motion, and 3D scene motions of rigid moving scenes.

CHAPTER 1
INTRODUCTION

1.1 Overview

We live in a three-dimensional (3D), fully dynamic world every day. The human perception system has the amazing abilities to sense the seamlessly changing 3D world through our embodied visual observations of the real world from only egocentric views. These abilities are crucial to build more autonomous robots or create more intelligent devices that can augment human capabilities. Autonomous robots need to walk, drive, fly, and interact with our real-world complex changing scenes. We require our future mixture reality devices to understand the human-level sensing of the world and behavior understanding of people around us. To empower our machines to achieve these abilities, one central question in perception is how we can represent the 3D changing world precisely and efficiently?

One solution is to represent the dynamic world as a time-varying dense continuous 3D motion field, termed as *scene flow* [1]. There are decades of researches in exploiting scene flow as a meaningful motion representation to analyze digital videos [2, 3] or as dense temporal correspondences to reconstruct the 3D world [4, 5, 6]. Several previous work applied scene flow to outdoor autonomous vechicles [7, 8, 9, 10] or indoor manipulation robot [11], which demonstrated scene flow can benefit the holistic scene understanding capabilities of the autonomous agent.

Computing scene flow from a moving camera subsumes several well-established computer vision problems. Vedula et al. [1] first introduce the term scene flow as optical flow in the 3D space. When projecting scene flow into the ego-centric views, the projected 2D vectors are *optical flow*. When depth is not known, estimating geometry involves *stereo estimation* from calibrated binocular pairs or temporal monocular views. In the rigidly

moving dynamic scenes, the 3D structure of scenes can be computed from a video sequence (up to scale) is a variant of the *structure-from-motion (SFM)*, if the extrinsics of the camera are calibrated. The camera motion or object motion can be recovered using the epipolar geometry if one of them is known. Accurately localizing the cameras in a dynamic scene is part of the SFM task and also widely known as *Visual Simultaneous Localization and Mapping (Visual SLAM)*.

The close connection of scene flow to its sub-problems in stereo, optical flow, motion segmentation and SFM have inspired many methods [12, 13, 14, 15, 16] to jointly optimizing the sub-tasks holistically for an accurate solution. While these methods achieved a significant progress in the accuracy of scene flow, benchmarked by the KITTI scene flow dataset [13], the scene flow remains as a pre-mature vision problem. The existing solution has not been as widely used as its sub-tasks in stereo, optical flow, SFM or SLAM in the real-world applications.

This thesis argues for importance to explore *fast*, *dense* and *accurate* scene flow approahes. Being dense and fast are essential properties to enable scene flow as a necessary and reliable representation to understand the dynamic scenes as human beings.

The Importance of Dense Representation: Many applications of scene flow require a *dense* representation, a measurement per pixel. Different from the classical SFM problem which focuses on reconstructing the 3D static scenes from the background region of the images, the scene motions about moving objects that have fewer measurements both per frame and over-time. To precisely and sufficiently represent such motions, it naturally requires the representation is dense to the limit in measurement quality.

Importance of Online Speed: In the modern applications of autonomous robots or mixed reality, the problem must be continuously solved *on-line* in *real-time*. It relates acutely to the challenges in human perception. Low-latency inference is an essential requirement to provide perception inputs for the agent to react and replan its high-level decision objective. In the modern computer vision application where the video input can stream as high

as 120Hz, a fast algorithm can more effectively leverage the sensing measurements which potentially reduce the difficulty in data association. However, existing optimization based scene flow algorithm often requires orders of magnitude more time in computation. Fast, accurate estimation of dense scene flow remains as an open problem with numerous challenges. The subtasks of the 3D scene motions in 2D are challenging problems in their own right. Reasoning about the 3D scene must still cope with a similar aperture problem as optical flow caused by motion ambiguity or textureless region in correspondence searching. Solving the scene flow in the absolute world coordinate, in addition, requires to address the ambiguities of scene motions and camera motion. From the views of an ego-centric moving robot or device, the ego-motion and object motions are naturally entangled in our observation. Classical approaches solve scene flow as an energy optimization from bottom-up spatiotemporal correlations in pixels e.g. total-variation[17], which are not able to provide accurate estimation tackling the aforementioned challenges. The recent approaches using holistic reasoning of its subtasks [15, 14] demonstrate a potential path to address these challenges using joint optimization, which, however, significantly sacrifices the inference speed for accuracy.

The recent significant progress in the data-driven approaches provides new opportunities towards solving scene flow more accurately and efficiently. The surge of learning approaches also come with new challenges in generalization. In this thesis, we will present an in-depth synthesis of classical optimization and deep learning to explore the potential directions to achieve fast, accurate, dense scene flow.

1.2 Thesis Contributions and Outline

The goal of this thesis to achieve *fast*, *accurate* and *dense* 3D motion estimations, which requires us rethinking both the motion representation as well as the algorithm. We focus on solving 3D dense motion field in entirely unconstrained environments from only limited views of ego-centric images. Through the entire thesis, we assume the camera intrinsics

are calibrated and known. The observations are from an ego-centric free moving camera. Our objective is to simultaneously recover camera ego-motion, the dense scene motion as well as the depth if the geometry is unknown.

The thesis will start from a brief historical review of the scene flow algorithm in *Chapter 2*. In each of our proposed work, we will take a deeper look at the most relevant work and discuss our connection and contribution to the overall progress in the community. In *Chapter 3*, we will briefly discuss the 3D geometry and optimization fundamentals.

We propose to explore an efficient and accurate scene flow algorithm from a joint perspective of least-square optimization and learning. Specifically, we present four approaches to tackle the problem and make the following contributions from Chapter 4 to Chapter 7.

***Chapter 4* Efficient Continuous Least-square Optimization of Planar Scene Flow:** We first propose a continuous least-square optimization method for solving dense 3D scene flow problems. As in recent work [15], we represent the dynamic 3D scene as a collection of rigidly moving planar segments. Rather than solving it using as a discrete-continuous optimization problem in existing work which leads to slow inference, we propose a purely continuous least-square formulation which can be solved more efficiently. Using a fine superpixel segmentation that is fixed a-priori, we present a factor graph formulation that decomposes the problem into photometric, geometric, and smoothing constraints. We find that with a high-quality correspondences [18], we can solve the nonlinear objective using the least-square method efficiently.

***Chapter 5* Learning Point-based Scene Flow:** The recent success using a data-driven approach to learn the estimation of 2D optical flow [19] is a promising direction to address scene flow efficiently using data. To recover scene flow from the 2D flow, we find the main challenge is the disambiguation of the camera motion from scene motion, which becomes more difficult as the amount of rigidity observed decreases. We propose to *learn* the rigidity of a scene in a supervised manner from an extensive collection of dynamic scene data, and directly infer a rigidity mask from two sequential images with depths. With

the learned rigidity network, we demonstrate we can estimate camera motion and projected scene flow using computed 2D optical flow and the inferred rigidity mask. For training and testing the rigidity network, we also provide a new semi-synthetic dynamic scene dataset (synthetic foreground objects with a real background). Our evaluation shows the proposed framework not only can outperform existing state-of-the-art scene flow estimation methods in challenging dynamic scenes in accuracy but also shows a promising direction to achieve real-time performance.

Chapter 6 **Learning Instance Scene Flow with Unrolled Optimization:** As an alternative to point-based scene flow, we propose to view scene flow estimation as a rigid instance motion alignment problem. In this work, we provide a modern synthesis of the classic inverse compositional algorithm for dense image alignment. We unroll a robust version of the inverse compositional algorithm and replace multiple components of this algorithm using more expressive models whose parameters we train in an end-to-end fashion from data. The proposed method can cope with challenging scenarios with noisy measurements and heavy occlusions. Moreover, it demonstrates a promising direction to seamlessly integrate learning into optimization, which can lead to a more robust, accurate optimization method with faster inference and also less prone to overfitting.

Chapter 7 **Learning to Recover Monocular Instance Scene Flow:** Inspired by the recent success in single-view depth estimation, we further study the scene flow observed from only two monocular consecutive views. Different from the existing work that learns scene flow as a composition of single-view image to depth regression and two-view images to optical flow, we propose to leverage motion-parallax and jointly estimate monocular scene flow using least-square optimization. Specifically, we propose to decompose the dynamic scene as rigid moving segments using instance segmentation method [20], and learn to reweight in optimized depth and 3D motion for each rigid instance. We learn to reweight the Gauss-Newton updates, which can ensure the motion and geometry consistency across views while overcoming the effect of outliers. Our evaluation shows the proposed method

can benefit from the progress in monocular depth and flow estimation while outperforming them with the integration of learning and optimization.

In *Chapter 8*, we summarize our contributions and insights gained during the experiments. We also provide an outlook to the important unsolved challenges and future opportunities in relevant tasks.

CHAPTER 2

SCENE FLOW REVIEW

3D motion estimation is a fundamental computer vision task that has a long history. S. Ullman [21] discusses 3D scene motion as a part of structure-from-motion with a particular assumption in the scene (rigidity) and camera (calibrated). Assuming the scene is piece-wise rigid, dynamic 3D motions can be calculated as an extension to structure-from-motion algorithms [22, 23]. Before the term scene flow being introduced in [1], Scene flow has also been extensively studied as motion-stereo [24, 25].

2.1 Scene Flow Classification

Vedula et al. [1] first defined three-dimensional motion estimation as scene flow and classify the 3D scene flow methods cross three different scenarios:

- having complete knowledge about the scene;
- knowing only stereo correspondence;
- no knowledge about the 3D structure.

The later two scenarios are often categorized as the stereo scene flow problem which requires jointly estimating 3D geometry and correspondence. The first scenario require pre-known geometry which can be acquired from depth sensor. We will separately discuss the relevant approaches.

Stereo Scene Flow: Estimate scene flow without 3D structure is the most challenging situation. Jointly estimation of disparity and flow has been shown to be beneficial to improve the accuracy for both-subtasks, which can be traced as early as the motion stereo work in

[24]. Huguet et al. [17] borrows a similar set-up to infer scene flow jointly in the two-stereo view setup and solve it as a joint variational inference of two stereo and optical flow problem. This set-up is now widely used as a standard in binocular scene flow. Valgaerts et al. [26] generalizes the framework with unknown camera transformation between the two views. Basha et al. [27] extends the joint variational inference into 3D by representing the structure as a point cloud, and estimate scene flow globally in the 3D world. Vogel et al. [15] presents the 3D scene using piece-wise rigid planes instead of individual points, and solve scene flow as a joint estimation as over-segmentation, plane assignment, and occlusion reasoning. In [16], they further extend the joint optimization to multi-frame scene flow, which demonstrated the benefits of holistic understanding scene flow.

Benchmark of rigid moving objects in the urban scenario: Menze et al. [13] focus on scene flow problem in the autonomous driving scenario and set up the KITTI Scene Flow benchmark for rigid scene motions estimation. The benchmark contains 200 stereo image pair for training and testing set each. They acquire the 3D motion ground truth by registering the 3D CAD vehicle model to the Lidar scans in each frame. Since then, it has benchmarked significant progress in scene flow research. There are several observations from the benchmark. First, exploiting scene understanding helps to improve scene flow accuracy. Ren et al. [14] jointly estimate semantic segmentation and scene flow via solving both tasks jointly as conditional random fields. Behl et al. [12] further incorporate the instance segmentation into scene flow optimization. Using the semantic and instance segmentation can better cope with the large flow displacement and moving object boundaries. Second, using multi-frame measurements [28, 16] shows favorable results compared with the two-frame approaches. Third, the learning-based methods can also achieve top performance [29, 30]. In general, these methods do not generalize directly to KITTI test set when training on synthetic scene flow data [31]. But they can conquer the KITTI test set by further finetuning on KITTI training data. Compared to optimization-based approaches, the learning-based methods are fast in inference time and has the potential to achieve real-time

performance. Ma et al. [32] specifically tackle the rigid instance motion by end-to-end learn the optical flow and stereo networks via a rigidity based unrolled optimization layers. This method is currently the top-performing method on KITTI. One drawback of the KITTI scene flow benchmark is that vehicle motions are the only moving objects in the scene. No nonrigid or articulated motions are included. The best performing methods often rely on the rigid motion hypothesis as regularization, which however may not demonstrate the progress in the general nonrigid scene flow.

RGBD Scene Flow: With the development of RGBD camera, directly using the depth information is attractive for approaches which requires fast computation for dense outputs. The depth camera provides an approximated geometry knowledge of the scene which simplifies the scene flow to a 3D correspondence problem. Several approaches [33, 34] extend the variational 2D optical flow directly to 3D scene flow using point cloud recovered from depth. Jaimez et al. [35] efficiently solves the dense RGB-D scene flow with a prime-dual method implemented in GPU. Sun et al. [36] regularizes the scene flow inference as finite 3D rigid moving objects, which can be relaxed as piece-wise smooth rigid moving objects in [37]. Quiroga et al. [38] proposes to represent an over-parameterized 6 DoF rigid transformation for each 3D point. They propose to model the scene as a rigid component induced by camera motion and a non-rigid transform as foreground, which inspires us to explore the connection of rigidity and scene flow in Chapter 5. Jaimez et al. [39] extends such separation by incorporating motion estimation from visual odometry. It uses two-stage rigid based clustering to decomposes the camera motion estimation and the non-rigid scene flow. It first clusters the background based on depth information and solves the visual odometry. Then it further clusters the foreground motions into different rigid moving pieces and solves them jointly as a piece-wise smooth rigid motion field. It is the first approach that can solve scene flow approximately in real-time. All the existing RGBD scene flow approaches are optimization-based, which requires certain rigid assumptions. Solving these nonlinear optimization problems suffer the local optima and may not work

in the wild. Besides using RGBD camera, there are several approaches estimate scene flow using point clouds from a LiDAR scanner in autonomous driving setting, including optimization-based work [40] and learning-based work [41, 42].

2.2 Concepts towards Fast Scene Flow

There are several paradigms in existing optimization-based approaches to reduce the computation for real-time applications. First, several approaches [43, 35, 44] limit the maximum displacement range for scene flow estimation. However, there is an obvious trade-off: these methods cannot deal with large displacements scene flow. Second, another commonly used paradigm is to decouple the scene flow constraints into sub-tasks and solve them sequentially, represented by [39, 28, 44]. In [44], they propose to decouple stereo and flow estimation, with each problem solved in parallel using GPU or FPGA. Jaimez et al. [39] iteratively solve visual odometry, motion segmentation and correspondence assocition. Taniai et al. [28] similarly solves these problem from a stereo setup sequentially. This solution can achieve near real-time RGBD scene flow computed only using CPU. Third, most of these approaches also utilizes parallel computing, such as GPU [43, 37, 45], FPGA [44], and multi-threaded CPU [39]. Recently, a number of learning-based scene flow approaches can achieve approximated real-time performance, which we will expand the discussion in the next section.

2.3 Learning-based Scene Flow

The recent progress in deep learning brings an alternative trend to motion estimation. Different from the classical approaches iteratively searching for a local optimal using optimization, learning-based approaches directly learn the mapping from the input to motion parameters using large amounts of data. The mapping is approximated via a feed-forward network which is usually fast with modern GPUs with highly optimized convolutional operations. Prominent examples include optical flow from two-views [19, 46, 47, 48] single

image to camera pose regression [49, 50, 51], image-based 3D object pose estimation [52, 53], relative pose and monocular depth prediction from two views [54, 55].

As a learning-based stereo scene flow baseline, Mayer et al. [31] proposes to refine the stacked stereo and optical flow via an additional feed-forward network. Recently, Jiang et al. [30] propose a shared encoder for stereo and optical flow estimation, which is the state-of-art learning-based scene flow method. Both of the methods are scene flow in reference coordinate, which do not require the global camera pose to be solved. In Chapter 5, we propose to learn the RGBD scene flow in the absolute world coordinate. Training such algorithms require a significant amount of data, which however is non-trivial to be acquired. Most of the existing scene flow approaches use random generated 3D object motions, e.g., FlyingChairs [19] and FlyingThings dataset in [31]. With domain randomization, the scene flow approaches can generalize surprisingly well to real-world scenes. However, existing datasets do not provide a realistic geometric environment with nonrigid motions. In Chapter 5, we propose a 3D scene aware rendering method with nonrigid moving humans.

Recently, there are many self-supervised learning approaches utilize view-synthesis via binocular consistency [56] and multi-frame consistency [57]. There are significant progress in self-supervised learning scene flow by enforcing multi-view constraints [58, 59], factorizing rigid and nonrigid motion [60, 61]. However, existing self-supervised approaches do not produce as accurate results as the supervised approaches [30, 32].

2.4 Other Relevant Approaches

Other researches in computer vision also have close relations with scene flow estimation, particularly in Non-rigid Structure from Motion (NRSfM) and dynamic reconstruction.

Non-rigid Structure-from-Motion : Derived from the Tomasi and Kanade's factorization technique [62], several approaches [63, 64, 22] extend it from a single rigid body transform to multi-body and non-rigid structures using the same orthographic projection. Similar to the scene flow problem, NRSfM is also an ill-posed problem that it solves time-varying

3D shapes observed from a few images at different viewpoints which requires additional priors as regularization[64, 65]. Dai et al. [66] proposed to factorize a low-rank model of non-rigidity which can recover both cameras and 3D shapes with no need of additional priors. However, its low-rank assumption can not generalize to complex sequences. Recent NRSfM approaches [67, 68] jointly estimate dense 3D structures and optical flow, which is a variant of monocular scene flow. However, no existing approaches tackle the problem from a learning perspective, which is the focus in Chapter 7.

Dynamic Reconstruction: Inspired by the success in KinectFusion [69], most existing dynamic reconstruction approaches [70, 4, 6, 71] extend it to using a similar frame-to-model tracking and are close to real-time tracking. DynamicFusion [6] estimates a set of sparse volumetric rigid transformations to align the moving object to the canonical model, termed as a warping field which is inherently a volumetric representation for scene flow. Fusion4D [70] achieves a dynamic scene reconstruction with multiple static cameras. Most of the dynamic reconstruction approaches are human-centric. With additional human template model, e.g. SMPL[72], the tracking can be more robust [71].

In the thesis, we will progressive study and discuss the related work in each chapter. The success in many of the methods inspired us to move forward towards the more general, fast and accurate dense scene flow estimation.

CHAPTER 3

FUNDAMENTALS

In this chapter, we will summarize the fundamental mathematical tools, computer vision concepts used throughout the thesis.

3.1 Two-view 3D Geometry

In this section we will first introduce some basic 3D geometric representations and the operations that will be used in the remainder of the thesis. We recommend readers to refer [73, 74] for a detailed tutorial about the 3D geometry particularly in context of the nonlinear optimization.

3.1.1 Camera Model

We suppose a pin-hole camera without distortion in all following work, with the intrinsic matrix $\mathbf{K}$, parameterized as $[f_x, f_y, c_u, c_v]$ with f_x, f_y as its focal length and c_u, c_v as its offset along the two axes. We define the pin-hle camera matrix as

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_u \\ 0 & f_y & c_v \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

We define a pixel $\mathbf{x} = [u, v] \in \mathbb{R}^2$. $\mathbf{p_x} = [\mathbf{p}_x, \mathbf{p}_y, \mathbf{p}_z] \in \mathbb{R}^3$ is the 3D point corresponding to $\mathbf{x}$ with depth $\mathbf{p}_z$. In some applications, we also define the pixel in the normalized image

coordinate $\mathbf{p}_{\mathbf{K}^{-1}\mathbf{x}} \in \mathbb{R}^3$ as:

$$\mathbf{p}_{\mathbf{K}^{-1}\mathbf{x}} = \begin{bmatrix} (u - c_u)/f_x \\ (v - c_v/c_v \\ 1 \end{bmatrix} \tag{3.2}$$

All the operations are defined in the homogeneous coordinate. The transformation from the 3D world to the image plane is a perspective projection. We define the perspective projection operation $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ and the inverse perspective projection operation $\pi^{-1} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ as

$$\pi(\begin{bmatrix} x \\ y \\ z \end{bmatrix}) : \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{z} \begin{bmatrix} x \\ y \\ z \end{bmatrix} ; \pi^{-1}(\begin{bmatrix} u \\ v \end{bmatrix}, z) : \begin{bmatrix} x \\ y \\ z \end{bmatrix} = z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{3.3}$$

The projected 2D image point in the homogenous coordinate is

$$\mathbf{x} = \pi(\mathbf{K}\mathbf{p}_{\mathbf{x}}) \tag{3.4}$$

and the inverse projection $\mathbf{p}_{\mathbf{x}} = \pi^{-1}(\mathbf{x}, \mathbf{p}_z) : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ as

$$\mathbf{p}_{\mathbf{x}} = \begin{bmatrix} \mathbf{p}_x \\ \mathbf{p}_y \\ \mathbf{p}_z \end{bmatrix} = \mathbf{p}_z \begin{bmatrix} \mathbf{p}_u \\ \mathbf{p}_v \\ 1 \end{bmatrix} = \mathbf{p}_z \begin{bmatrix} (u - c_u)/f_x \\ (v - c_v)/f_y \\ 1 \end{bmatrix} = \mathbf{p}_z \mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{3.5}$$

Jacobian: In a number of the optimization problems discussed in the thesis, we will derive the Jacobian $\mathbf{J}_\pi$ of the pin-hole projection by the coordinate of the point w.r.t. the camera,

which is

$$\mathbf{J}_\pi = \begin{bmatrix} f_x/\mathbf{p}_z & 0 & f_x\mathbf{p}_x/\mathbf{p}_z^2 \\ 0 & f_y/\mathbf{p}_z & f_y\mathbf{p}_y/\mathbf{p}_z^2 \end{bmatrix} = \begin{bmatrix} f_x/\mathbf{p}_z & 0 & f_x\mathbf{p}_u/\mathbf{p}_z \\ 0 & f_y/\mathbf{p}_z & f_y\mathbf{p}_v/\mathbf{p}_z \end{bmatrix} \tag{3.6}$$

3.1.2 3D Rigid Body Transform

We define the 3D rigid body transform of a 3D point $\mathbf{p_x}$ as

$$\mathbf{p}'_\mathbf{x} = \mathbf{R}\mathbf{p}_\mathbf{x} + \mathbf{t} \tag{3.7}$$

where $\mathbf{R}$ is a 3x3 orthonomal rotation matrix and $\mathbf{t} \in \mathbb{R}^3$ is the translation vector. The rotation matrix is an overparameterization of the underlying group which only has 3 degree of freedom. Although it is easy calculate the point transformation using the matrix representation, the overparameterized form is not a smooth manifold and cannot be directly estimated without incorporating further constraint.

Lie Group Representation

A minimal representation of 3D rigid transform can be derived using Lie group theory. A Lie group is a smooth manifold, where the group operations (multiplication and inversion) are smooth maps. In this thesis, we will primary focus on the *Special Euclidean Group* $\mathbf{SE3}$ for the 6DoF general rigid transform and the *Special Orthogonal Group* $\mathbf{SO3}$ for the general 3DoF rotation transform.

The general 6DoF rigid transform matrix can be represented as a Special Euclidean Group $\mathbf{SE3}$ as

$$\left\{ \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \middle| \mathbf{R} \in \mathbf{SO3}, \mathbf{t} \in \mathbb{R}^3 \right\} \tag{3.8}$$

where the rotation is represented using Special Orthogonal Group $\mathbf{SO3}$, a three-dimensional manifold embeded within a 9-dimensional ambient space.

The minimal representation of $\mathbf{SE3}$ and $\mathbf{SO3}$ can be defined using *Lie Alegbra*. There is an exact, surjective mapping from Lie alegbra to the corresponding Lie Group using matrix exponential

$$\mathbf{R}(\boldsymbol{\omega}) = \exp(\hat{\boldsymbol{\omega}}) : \mathfrak{so}(3) \rightarrow \mathbf{SO3} \tag{3.9}$$

$$\mathbf{T}(\boldsymbol{\xi}) = \exp(\hat{\boldsymbol{\xi}}) : \mathfrak{se}(3) \rightarrow \mathbf{SE3} \tag{3.10}$$

where the operator $\hat{}$ defines the mapping from vector space of $\boldsymbol{\omega} \in \mathbb{R}^3, \boldsymbol{\xi} \in \mathbb{R}^6$ to their lie algebra $\mathfrak{so}(3), \mathfrak{se}(3)$. The matrix representation to the Lie algebra can refer to matrix logarithm vice versa.

Lie Algebra of SO3 and SE3 : Lie algebra defines the structure of the tangent space around the identity. It can be represented as linear combinations of its generator matrices. Consider the following generator matrices

$$\left\{ \begin{array}{llll} G_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, & G_2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, & G_3 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \\ G_4 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, & G_5 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, & G_6 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \right\}$$

The Lie alegbra $\mathfrak{so}(3)$ is the vector space of a skew-symmetric matrices $\hat{\boldsymbol{\omega}} = [\boldsymbol{\omega}]_\times$

$$\hat{\boldsymbol{\omega}} = [\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_x \\ \omega_z & 0 & -\omega_y \\ -\omega_x & \omega_y & 0 \end{bmatrix} \tag{3.11}$$

$$= \omega_x G_1 + \omega_y G_2 + \omega_z G_3 \tag{3.12}$$

corresponding to a rotation around each axis in x, y, z respectively. With the generator matrices, we can represent the Lie algebras corresponding to the introduced Lie groups $\mathbf{SO}(3)$, $\mathbf{SE}(3)$ as

$$\mathfrak{so}(3) = \left\{ \omega_x G_1 + \omega_y G_2 + \omega_z G_3 | (\omega_x, \omega_y, \omega_z) \in \mathbb{R}3 \right\} \tag{3.13}$$

$$\mathfrak{se}(3) = \left\{ \hat{\boldsymbol{\omega}} + v_x G_4 + v_y G_5 + v_z G_6 | \boldsymbol{\omega} \in \mathfrak{so}(3), (v_x, v_y, v_z) \in \mathbb{R}^3 \right\} \tag{3.14}$$

In this thesis, we will represent the 3D rigid transform $\boldsymbol{\xi} \in \mathfrak{se}(3)$ using Lie algebra in default. The derivatives with respect to the underlying motion types are intuitive to use in the context of gradient-based optimization. Refer to [74] for more details of the Lie Group and its derivatives.

SO3 to axis angle: A rotation can also be represented by an unit rotation axis $\bar{\boldsymbol{\omega}}$ and an angle θ in magnitude, or equivalently by a 3D vector $\boldsymbol{\omega} = \theta\bar{\boldsymbol{\omega}}$. In the canonical representation of $\mathbf{SO3}$ in (3.9), $\boldsymbol{\omega}$ is known as the canonical or exponential coordinates, which is equivalent to the axis-angle representation for rotations $\boldsymbol{\omega}$. The rotation matrix $\mathbf{R}(\boldsymbol{\omega})$ can be derived from the axis angle representation known as the *Rodriguez's formula*

$$\mathbf{R}(\bar{\boldsymbol{\omega}}, \theta) = \mathbf{I} + \sin(\theta)[\bar{\boldsymbol{\omega}}]_\times + (1 - \cos(\theta))[\bar{\boldsymbol{\omega}}]_\times^2 \tag{3.15}$$

$$= \mathbf{R}(\boldsymbol{\omega}) = \mathbf{I} + \frac{\sin(\theta)}{\theta}\hat{\boldsymbol{\omega}} + \frac{(1 - \cos(\theta))}{\theta^2}\hat{\boldsymbol{\omega}}^2 \tag{3.16}$$

(3.16) is the close-form expoential map of $\mathbf{SO3}$ using Rodriguez's formula.

The axis angle representation is not unique, since we can always add a multiple of $360°$ (2π radians) to and get the same rotation matrix. As well, ($\bar{\boldsymbol{\omega}}$, θ) and ($-\bar{\boldsymbol{\omega}}$, $-\theta$) represent the same rotation. However, for small rotations, or local perturbation, axis angle is an excellent choice. In particular, the Rodriguez's formula can be simplified when $\boldsymbol{\omega}$ is small

$$\mathbf{R}(\boldsymbol{\omega}) \approx \mathbf{I} + \hat{\boldsymbol{\omega}} \approx \begin{bmatrix} 1 & -\omega_z & \omega_x \\ \omega_z & 1 & -\omega_y \\ -\omega_x & \omega_y & 1 \end{bmatrix} \tag{3.17}$$

From (3.17), we can see the derivative of rotation $\mathbf{R}(\boldsymbol{\omega})$ is simply an unit skew-symmetric matrix. In the gradient based optimization approach, we often estimate the local incremental update in the continuous optimization with the small perturbation assumption.

We define an incremental transformation as tracing out a geodesic curve the group manifold along a certain tangent vector $\delta\boldsymbol{\xi} \in \mathfrak{se}(3)$. In the optimization problem, we are often interested to know the perturbed function with a small increment $\boldsymbol{\epsilon}$ around the variable pose $\boldsymbol{\epsilon}(\boldsymbol{\xi}) \in \mathbb{R}^3$ or the point $\boldsymbol{\epsilon}(\mathbf{p}) \in \mathbb{R}^3$. For tangent vector $\delta\boldsymbol{\xi}$, the Jacobian matrix $\mathbf{J}_{\boldsymbol{\xi}}(\mathbf{p})$ of the transformation $\mathbf{T}(\delta\boldsymbol{\xi})\mathbf{p}$ is

$$\mathbf{J}_{\boldsymbol{\xi}}(\mathbf{p}) = \begin{bmatrix} [-\mathbf{p}]_\times & \mathbf{I}_{3\times 3} \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{p}_z & -\mathbf{p}_y & 1 & 0 & 0 \\ -\mathbf{p}_z & 0 & \mathbf{p}_x & 0 & 1 & 0 \\ \mathbf{p}_y & -\mathbf{p}_x & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.18}$$

$$\mathbf{J}_{\mathbf{p}}(\boldsymbol{\xi}) = \mathbf{T}(\boldsymbol{\xi}) \begin{bmatrix} \mathbf{I}_{1\times 3} \\ 0 \end{bmatrix} = \mathbf{R}(\boldsymbol{\xi}) \tag{3.19}$$

Euler Angles

Another way to represent 3D orientation is to use three angles $\Theta = [\alpha, \beta, \gamma]$ as three sequential rotations $\mathbf{R}(\alpha, \beta, \gamma) = \mathbf{R}_x(\alpha)\mathbf{R}_y(\beta)\mathbf{R}_z(\gamma)$ around certain axis. Note in the construction of Euler Angles, the ordering of the rotations along the axis matters.

This representation is often used in the navigation task. And it is also commonly used as rotation representation for pose regression tasks using convolutional networks [75, 60, 57]. Euler Angles representation is prone to the "Gimbal Lock" issue. In specific configurations, the system loses a degree of freedom, and Jacobian will lose one rank, which makes this representation unsuited for continuous optimization. In applications using two consecutive frames from a video stream, gimbal locks unlikely occur. In chapter Chapter 5, we train a neural network to regress from image to Euler Angles directly.

Perspective Projection Warping

Given a rigid body transform $\mathbf{T}(\boldsymbol{\xi})$ with $\boldsymbol{\xi} \in \mathfrak{se}(3)$, we define $\mathcal{W}_\pi(\mathbf{p}) : \mathbb{R}^3 \to \mathbb{R}^2$ is the warping function that projects the 3D point $\mathbf{p} \in \mathbb{R}^3$ to a 2D point $\mathbf{x}' \in \mathbb{R}^2$:

$$\mathbf{x}' = \mathcal{P}_\pi(\mathbf{p}) = \pi(\mathbf{K}\,\mathbf{T}(\boldsymbol{\xi})\,\mathbf{p}) \tag{3.20}$$

We can derive the perturbation as Jacobian function through chain-rule using (3.6), (3.18) and (3.19). The Jacobian of the warping function w.r.t. to the point $\mathbf{J}_{\mathcal{P}_\pi(\mathbf{p})}$ is

$$\mathbf{J}_{\mathcal{P}_\pi(\mathbf{p})}(\mathbf{x}') = \mathbf{J}_\pi(\mathbf{p}')\Big|_{\mathbf{p}'=\mathbf{T}(\boldsymbol{\xi})\mathbf{p}}\,\mathbf{J}_{\mathbf{p}}(\boldsymbol{\xi}) \tag{3.21}$$

$$= \begin{bmatrix} f_x/\mathbf{p}'_z & 0 & -f_x\mathbf{p}'_u/\mathbf{p}'_z \\ 0 & f_y/\mathbf{p}'_z & -f_y\mathbf{p}'_v/\mathbf{p}'_z \end{bmatrix} \mathbf{R}(\boldsymbol{\xi}) \tag{3.22}$$

where $\mathbf{p}' = \mathbf{T}(\boldsymbol{\xi})\mathbf{p}$. The Jacobian w.r.t. to the local motion $\mathbf{J}_{\mathbf{W}_\pi(\boldsymbol{\xi})}$ is

$$\mathbf{J}_{\mathcal{P}_\pi(\boldsymbol{\xi})}(\mathbf{x}') = \mathbf{J}_\pi(\mathbf{p}')\Big|_{\mathbf{p}'=\mathbf{T}(\boldsymbol{\xi})\mathbf{p}} \mathbf{J}_{\boldsymbol{\xi}}(\boldsymbol{\epsilon}(\boldsymbol{\xi}), \mathbf{p}) \tag{3.23}$$

$$= \begin{bmatrix} -f_x\mathbf{p}'_u\mathbf{p}'_v & f_x(1+\mathbf{p}'^2_u) & -f_x\mathbf{p}'_v & f_x/\mathbf{p}'_z & 0 & -f_x\mathbf{p}'_u/\mathbf{p}'_z \\ -f_y(1+\mathbf{p}'^2_v) & -f_y\mathbf{p}'_u\mathbf{p}'_v & f_y\mathbf{p}'_u & 0 & f_y/\mathbf{p}'_z & -f_y\mathbf{p}'_v/\mathbf{p}'_z \end{bmatrix} \tag{3.24}$$

Perspective Image to Image Warping

Given the 2D point $\mathbf{x} \in \mathbb{R}^2$ and its depth $bp_z \in \mathbb{R}$, we define the image to image rigid warping as $\mathcal{W}_\pi(\mathbf{p}) : \mathbb{R}^3 \to \mathbb{R}^2$

$$\mathbf{x}' = \mathcal{W}_\pi(\mathbf{x}, \mathbf{p}_z) = \pi(\mathbf{K}\,\mathbf{T}(\boldsymbol{\xi})\,\mathbf{K}^{-1}\mathbf{x}) \tag{3.25}$$

The image to image warping is more widely The Jacobian can be similarly defined through chain-rule:

$$\mathbf{J}_{\mathcal{W}_\mathbf{x}(\mathbf{p}_z)}(\mathbf{x}') = \mathbf{J}_{\mathcal{P}_\pi(\mathbf{p})}\Big|_{\mathbf{p}=\mathbf{K}^{-1}\pi^{-1}(\mathbf{x},\mathbf{p}_z)} \frac{\partial \mathbf{p}}{\partial \mathbf{p}_z} \tag{3.26}$$

$$= \begin{bmatrix} f_x/\mathbf{p}'_z & 0 & -f_x\mathbf{p}'_u/\mathbf{p}'_z \\ 0 & f_y/\mathbf{p}'_z & -f_y\mathbf{p}'_v/\mathbf{p}'_z \end{bmatrix} \mathbf{R}(\boldsymbol{\xi}) \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \tag{3.27}$$

$$\mathbf{J}_{\mathcal{W}_\mathbf{x}(\boldsymbol{\xi})}(\mathbf{x}') = \mathbf{J}_{\mathcal{P}_\pi(\boldsymbol{\xi})}(\mathbf{x}') \tag{3.28}$$

3.2 Nonlinear Optimization

Nonlinear Least-Square Objective: In this thesis, we discuss the optimization problem generally has a least-square form, i.e., can be written as

$$\underset{\boldsymbol{\xi}}{\operatorname{argmin}}\, \mathbf{r}^T(\boldsymbol{\xi})\, \mathbf{r}(\boldsymbol{\xi}) \tag{3.29}$$

where $\boldsymbol{\xi}$ is the variable to be estimation and $\mathbf{r} \in \mathbb{R}^{N\times 1}$ is the residual vector for N number of measurements. When $\delta\boldsymbol{\xi}$ contains parameter in a nonlinear group, e.g. $\mathbf{SO3}$ or $\mathbf{SE3}$, (3.29) is a nonlinear least-square optimization problem. One assumption in the least-suqare estimator is that the noise corrupting the data is of zero mean, which yields an unbiased parameter estimate.

A standard method solve the nonlinear least-squares problems via using an iterative method solving a linearised system starting from a suit-able initial estimate. In this section, we will start from the Gauss-Newton method in Section 3.2.1, as well as its regularizated form Section 3.2.2, particular the well-known extension Levenberg-Marquardt algorith. Then we will introduce robust M-estimator in Section 3.2.3 which robustifies the nonlinear optimization problem as the well-known iteratively reweighted least-square (IRLS) optimization in Section 3.2.4. In the end, we will briefly discuss the connection of least-square optimization to the factor graph representation in Section 3.2.5, which can be an effective tool in least-square problem modelling. We recommend the readers to read [76] for a comphrensive discussion about nonlinear optimization using factor graph.

3.2.1 Gauss-Newton Method

The general iterative method solves a quadratic approximation to (3.29) iteratively. At iteration k, we minimizes the residual $\mathbf{r}_k$ around an initial estimation $\boldsymbol{\xi}_0$ with a local perturbation $\boldsymbol{\xi}_0$.

$$\underset{\boldsymbol{\xi}}{\operatorname{argmin}} ||\mathbf{r}_k(\boldsymbol{\xi}_0 + \delta\boldsymbol{\xi})||_2^2 \tag{3.30}$$

The Gauss-Newton algorithm linearizes the residual $\mathbf{r}_k$, using a first-order Taylor expansion

$$||\mathbf{r}_k(\boldsymbol{\xi}_0 + \delta\boldsymbol{\xi})||_2^2 = ||\mathbf{r}_k^T + \mathbf{J}\delta\boldsymbol{\xi}||_2^2 \tag{3.31}$$

$$= \mathbf{r}_k^T\mathbf{r}_k + 2\boldsymbol{\xi}^T\mathbf{J}^T\mathbf{r}_k + \boldsymbol{\xi}^T\mathbf{J}^T\mathbf{J}\boldsymbol{\xi} \tag{3.32}$$

with simplified notation $\mathbf{r}_k = \mathbf{r}_k(\boldsymbol{\xi}_0), \mathbf{J} = \mathbf{J}_{\delta\boldsymbol{\xi}}(\boldsymbol{\xi}_0)$. The minimum of the quadratic function is equal to the solution of the following linear system by setting the derivative of the quadratic function to zero

$$(\mathbf{J}^T\mathbf{J})\delta\boldsymbol{\xi} = -\mathbf{J}^T\mathbf{r}_k \tag{3.33}$$

The incremental update $\delta\boldsymbol{\xi}$ is thus solved

$$\delta\boldsymbol{\xi} = -(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{r}_k \tag{3.34}$$

$$= -\mathbf{J}^+\mathbf{r}_k \tag{3.35}$$

where $\mathbf{J}^+$ is the Moore-Penrose pseudoinverse of $\mathbf{J}$. When the *approximate Hessian* matrix $\mathbf{J}^T\mathbf{J}$ is full-rank, there is an unique solution. The full-rank condition also indicates all paramters are full observable. The solution is updated by

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k + \delta\boldsymbol{\xi} \tag{3.36}$$

Manifold optimization: When optimizing on a general manifold, we define $\oplus$ as a generalized addition operator for the *Retraction* operator $\mathcal{R}(\boldsymbol{\xi}) : \mathcal{M} \times \mathbb{R}^n \to \mathcal{M}$ such that

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k \oplus \delta\boldsymbol{\xi} = \mathcal{R}_{\boldsymbol{\xi}}(\delta\boldsymbol{\xi}) \tag{3.37}$$

For the vector space $\mathbb{R}^n$, the retration is simply vector addition. For Lie group, the most used retraction is the exponetial map, although other retraction operator can also be defined. As defined in (3.1.2), the exponential map allows us to define a mapping from the local coodinates $\boldsymbol{\xi}$ back to a neighbourhood around the current estimated transform $\mathbf{T}(\boldsymbol{\xi}_k)$.

$$\boldsymbol{\xi}_k \oplus \delta\boldsymbol{\xi} = \mathbf{T}_k(\boldsymbol{\xi}_k) \cdot \exp(\hat{\boldsymbol{\xi}}) \tag{3.38}$$

Matrix factorization: When the approximated Hessian is sparse, Cholesky factorization is most used to solve (3.33), with $\mathbf{J}^T\mathbf{J} = \mathbf{L}^T\mathbf{L}$ where $\mathbf{L}$ is upper triangular. The solving $\boldsymbol{\xi}$ is a two-step back-substitution for $\mathbf{L}$ and $\mathbf{L}^T$ sequentially. Other than Cholesky factorization, QR and SVD factorization can also be used. Both QR and SVD factorization are significantly slower than cholesky factorization for sparse matrix. In certain cases, QR factorization has better numerical stablity. For sparse matrices factorization in this thesis, we use Cholesky factorization in default. To calculate the inverse of dense matrix, we use SVD factorization and simultaneously check the numerical stability during the factorization.

The limitation: In many cases, the approximated Hessian matrix $\mathbf{J}^T\mathbf{J}$ can be fully observable but ill-conditioned. In this thesis, we will often tackle the motion estimation by alignment photometric residuals. There will be two major issues. First, when minimizing the photometric residuals in textureless regions or along the boundary, the Jacobian along some parameter dimension has more significant magnitude[77]. Second, the quadratic approximation to the nonlinear objective is only meaningful if start from a proper initialization. When the estimated motion is substantial, or the measurements are noisy, this assumption can often break. Traditional methods often use the trust-region method to relax the quadratic approximation and the robust M-estimator to handle the influence of the outliers. We will briefly introduce them in the following context. In Chapter 6, of this thesis, we also propose a new learning-based paradigm to handle these issues.

3.2.2 Trust-Region Method

Gradient descent is an alternative solution to use a quadratic approximation. For sufficiently small step-size and under certain conditions, the gradient descent step is guaranteed to converge to a local minimum. However, compared to quadratic method, e.g. Gauss-Newton method, gradient descent is slow to converge, which is not suited for methods which require fast online optimization.

The Levenberg algorithm: The Levenberg-Marquardt algorithm is adaptively "interpolating" between Gauss-Newton and gradient descent steps via a dampening parameter λ

$$\Delta\boldsymbol{\xi} = \mathbf{H}^{-1}\,\mathbf{J}^{T}\,\mathbf{r}_{k} \tag{3.39}$$

$$\mathbf{H} = (\mathbf{J}^{T}\mathbf{J} + \lambda\,\mathbf{I}) \tag{3.40}$$

where $\mathbf{I}$ is a diagonal identity matrix and λ is the scalar value for damping. If λ is small, $\mathbf{J}^{T}\mathbf{J}$ will dominate the regularized approximated Hessian $\mathbf{H}$ and the (3.39) is close to a Gauss-Newton update. If λ is large, the diagonal damping $\lambda\mathbf{I}$ will dominate $\mathbf{H}$ and the computed step will be close to a gradient descent step. Step-size is $1/\lambda$ scaled with the curvature along each dimension to avoid slow convergence along dimensions with a low gradient.

The Levenberg-Marquardt algorithm: A more popular variant of the Levenberg Algorithm is the Levenberg-Marquardt formulation. It replaces the identity damping by the diagonal entries of the approximated Hessian matrix which helps to achieve faster convergence

$$\mathbf{H} = (\mathbf{J}^{T}\mathbf{J} + \lambda\,\mathrm{diag}(\mathbf{J}^{T}\mathbf{J})) \tag{3.41}$$

3.2.3 M-estimator

One caveat of the least-square estimator is the influence of the outliers. One outlier deviating from the correct solution may significantly impact the final estimation. As stated before, the least-squares estimators assume that the noise corrupting the data is of zero-mean Gaussian. If all the point measurements are independent, each point has independent variance which associate to its weight. If the noise variance is known, a minimum-variance parameter estimate can be obtained by choosing appropriate weights on the data. However, measuring the noise variance is often not feasible.

One popular technique to compensate for the influence of the outliers is the so-called *M-estimators*. Suppose r_i is the ith element of the residual vector $\mathbf{r} = [r_1, r_2, ..., r_n]$. Instead of minimizing the squared residuals $\sum_i r_i^2$, the M-estimator minimizes the following objective

$$\sum_i \rho(r_i) \tag{3.42}$$

The influence function: The Jacobian corresponds to the residual r_i is thus:

$$\mathbf{J}(r_i) = \phi(r_i)\frac{\partial \rho(r_i)}{\partial \boldsymbol{\xi}} \tag{3.43}$$

$$\phi(r_i) = \frac{\rho(r_i)}{r_i} \tag{3.44}$$

where the derivative function $\phi(\cdot)$ is the *influence function*. The infuence function measures the infuence of a datum on the value of the parameter estimate. For example, for the square error objective with $\phi(x) = x^2/2$, the infuence function $\phi(x) = x$ increases linearly with the size of its error, whichs means the inlier and outlier influence the estimtaor equally.

The weight function: We further expand (3.43) as

$$\mathbf{J}(r_i) = \omega(r_i) r_i \frac{\partial \rho(r_i)}{\partial \boldsymbol{\xi}} \tag{3.45}$$

$$\omega(r_i) = \frac{\phi(r_i)}{r_i} \tag{3.46}$$

where $\omega(\cdot)$ is the so-called the *weight function* of ρ. This is exactly equivalent to the following reweighted squared error as

$$\sum_i \omega(r_i) r_i^2 \tag{3.47}$$

Table 3.1: **A few commonly used M-estimators.** We enumerate a few error norms $\rho(x)$, the influence function $\phi(x)$ and the weight functions $\omega(x)$.

Type	Norm $\rho(x)$	Influence $\phi(x)$	Weight $\omega(x)$
L_2	$x^2/2$	x	1
L_1	$\lvert x\rvert$	1	0
L_p	$\lvert x\rvert^v/v$	$\text{sgn}(x)\lvert x\rvert^{v-1}$	$\lvert x\rvert^{v-2}$
Cauchy	$c^2/(2\log(1+(x/c)^2))$	$x/(1+(x/c))^2$	$1/(x/c)^2$
German-McClure	$x^2/(2(1+x^2))$	$x/(1+x^2)^2$	$1/(1+x^2)^2$
Huber $\begin{cases}\text{if} & \lvert x\rvert \leqslant k \\ \text{if} & \lvert x\rvert > k\end{cases}$	$\begin{cases}x^2/2 \\ k(\lvert x\rvert - k/2)\end{cases}$	$\begin{cases}x \\ k\text{sgn}(x)\end{cases}$	$\begin{cases}1 \\ k/\lvert x\rvert\end{cases}$
Tuckey $\begin{cases}\text{if} & \lvert x\rvert \leqslant c \\ \text{if} & \lvert x\rvert > c\end{cases}$	$\begin{cases}\frac{c^2}{6}(1-[1-(x/c)^2]^3) \\ c^2/6\end{cases}$	$\begin{cases}x[1-(x/c)^2]^2 \\ 0\end{cases}$	$\begin{cases}[1-(x/c)^2]^2 \\ 0\end{cases}$

or its vector form as

$$\mathbf{r}^T \mathbf{W} \mathbf{r} \tag{3.48}$$

where $\mathbf{W}$ is a diagonal matrix with each diagonal entry at row i is ω_i.

The robust M-estimator requires the the influence function $\phi(x)$ is bounded and the individual $\rho(x)$ function is convex in the variable x. In practice, there are multiple choices for $\rho(x)$ depending on the estimated distributions of the datum. A few commonly used M-estimators are listed in Table 3.1. A more comprehensive discussion of classical Robust M-estimators can be found in [78].

Choosing the right distribution of the datum can be a non-trivial task. J. Barron [79] introduces a general and adaptive robust loss function that can subsume the commonly used

M-estimator in one learnable function. The enumerated classical robust mestimator in Table 3.1 only depends on point residual, although higher order information may seem helpful to measure the error distributions. In Chapter 6, we will revisit the robust M-estimator in the context of image alignment and explore the solution to cope with its limitations.

3.2.4 Iteratively Reweighted Least-Squares Optimization

Using M-estimator, we minimize the following general robust least-squares objective

$$\operatorname{argmin} \mathbf{r}^T(\boldsymbol{\xi})\, \mathbf{W}\, \mathbf{r}(\boldsymbol{\xi}) \tag{3.49}$$

with update at each step as

$$\Delta\boldsymbol{\xi} = (\mathbf{J}^T\mathbf{W}\mathbf{J} + \lambda\, \text{diag}(\mathbf{J}^T\mathbf{W}\mathbf{J}))^{-1}\mathbf{J}^T\mathbf{W}\, \mathbf{r}_k \tag{3.50}$$

Algorithm 1 summarizes the iterative update steps.

Algorithm 1: The Iterative Reweighted Least-Squares Algorithm

while $||\mathbf{r}_k||_2^2 > \epsilon$ **do**
  $\mathbf{r}_k = \mathbf{r}(\boldsymbol{\xi}_k)$;
  $\mathbf{J} = \frac{\partial \mathbf{r}(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}$;
  adjust λ ;
  $\delta\boldsymbol{\xi} = -(\mathbf{J}^T\mathbf{W}\mathbf{J} + \lambda\, \text{diag}(\mathbf{J}^T\mathbf{W}\mathbf{J}))^{-1}\mathbf{J}^T\mathbf{W}\, \mathbf{r}_k$;
  $\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k \circ \delta\boldsymbol{\xi}$
end

3.2.5 Least-Square Optimization as a Factor Graph

There is a tight connection of non-linear least-square problem to factor graph [80]. A factor graph is a probabilistic graph model, which represents a joint probability distributions of all factors

$$p(x) \propto \prod_i p_i(x_i), \tag{3.51}$$

where $x_i \subseteq x$ is subset of variables involved by factor i, $p(x)$ is the overall distribution of the factor graph, and $p_i(x_i)$ is the distribution of each factor. The maximum a posteriori (MAP) estimate is

$$x^* = \underset{x}{\operatorname{argmax}}\, p(x) \tag{3.52}$$

$$= \underset{x}{\operatorname{argmax}} \prod_i p_i(x_i). \tag{3.53}$$

Suppose each factor has Gaussian distribution on $f_i(x_i)$ with covariance Σ_i,

$$p_i(x_i) \propto \exp\big(-\frac{1}{2} \parallel f_i(x_i) \parallel^2_{\Sigma_i} \big), \tag{3.54}$$

the MAP inference has

$$x^* = \underset{x}{\operatorname{argmax}} \prod_i p_i(x_i) \tag{3.55}$$

$$= \underset{x}{\operatorname{argmax}} \log\big(\prod_i p_i(x_i) \big), \tag{3.56}$$

$$= \underset{x}{\operatorname{argmin}} \prod_i -\log\big(p_i(x_i)\big) \tag{3.57}$$

$$= \underset{x}{\operatorname{argmin}} \sum_i \parallel f_i(x_i) \parallel^2_{\Sigma_i} . \tag{3.58}$$

The MAP inference problem in (3.58)) is converted to the same non-linear least square

optimization problem in equation as (3.36), which can be solved following the same steps in (3.36).

There are several advantages using factor graph to model non-linear least square problems. Factor graphs encode probabilistic nature of the problems, and easily indicate the underlying sparsity of the optimization problem since for most (if not all) factors x_i are very small sets of the overall variables. In Chapter 4, we use factor graph as the representation and solve the MAP inference as a sparse least-square optimization problem.

CHAPTER 4

A LEAST-SQUARE SOLUTION TO SCENE FLOW

Summary

We propose a continuous optimization method for solving dense 3D scene flow problems from stereo imagery. As in recent work, we represent the dynamic 3D scene as a collection of rigidly moving planar segments. The scene flow problem then becomes the joint estimation of pixel-to-segment assignment, 3D position, normal vector and rigid motion parameters for each segment, leading to a complex and expensive discrete-continuous optimization problem. In contrast, we propose a purely continuous formulation which can be solved more efficiently. Using a fine superpixel segmentation that is fixed a-priori, we propose a factor graph formulation that decomposes the problem into photometric, geometric, and smoothing constraints. We initialize the solution with a novel, high-quality initialization method, then independently refine the geometry and motion of the scene, and finally perform a global non-linear refinement using Levenberg-Marquardt. We evaluate our method in the challenging KITTI Scene Flow benchmark, ranking in third position at the time of submission, while being 3 to 30 times faster than the top competitors.

4.1 Introduction

Solving scene flow is traditionally casted as an energy optimization problem. Several recent approaches assume the 3D world over time is composed of rigid moving planes and achieved impressive accurate scene flow estimation [15, 13, 16]. In these approaches, the scene is represented using planar segments which are assumed to have consistent spatial and temporal motions. The scene flow problem is then posed as a discrete-continuous optimization problem which associates each pixel with a planar segment, each of which has continuous rigid 3D motion parameters to be optimized.

These approaches cast the entire scene flow estimation into a joint discrete optimization problem with other relevant tasks. However, joint inference in this space is both complex and computationally expensive. Menze and Geiger [13] partially address this by parameter-sharing between multiple planar segments, by assuming the existence of a finite set of moving objects in the scene. They solve the candidate motion of objects with continuous optimization, and use discrete optimization to assign the label of each object to each superpixel. However, this assumption does not hold for scenes with non-rigid deformations. Piece-wise continuous planar assumption is not limited to 3D description.

In contrast to this body of optimization approaches, we posit that it is better to solve for the scene flow in the continuous domain, which is faster in inference without losing accuracy. We adopt the same rigid planar representation as [15], but solve it more efficiently with high accuracy. Instead of reasoning about discrete labels, we use a fine superpixel segmentation that is fixed a-priori, and utilize a robust nonlinear least-squares approach to cope with occlusion, depth and motion discontinuities in the scene. A central assumption is that once a fine enough superpixel segmentation is used as a priori, there is no need to jointly optimize the superpixel segmentation within the system. The rest of the scene flow problem, being piecewise continuous, can be optimized entirely in continuous domain. A good initialization is obtained by leveraging *DeepMatching* [18]. We achieve fast inference

by using a sparse nonlinear least squares solver and avoid discrete approximation. To utilize Census cost for fast robust cost evalution in continuous optimization, we proposes a differentiable Census-based cost, similar to but not same as the approach in [81].

Contributions: In summary, this work makes the following contributions:

1. We propose a factor-graph formulation of the scene flow problem that exposes the inherent sparsity of the problem, and use a state of the art sparse solver that directly optimizes over the manifold representations of the continuous unknowns. Compared to the same representation in [15], we achieve better accuracy and faster inference.

2. Instead of directly solving for all unknowns, we propose a pipeline to decompose geometry and motion estimation. We show that this helps us cope with the highly nonlinear nature of the objective function.

3. As initialization is crucial for nonlinear optimization to succeed, we use the DeepMatching algorithm from [18] to obtain a semi-dense set of feature correspondences from which we initialize the 3D motion of each planar segment. As in [13], we initialize planes from a restricted set of motion hypotheses, but optimize them in the continuous domain to cope with non-rigid objects in the scene.

4.2 Scene Flow Analysis

We follow [15] in assuming that our 3D world is composed of locally smooth and rigid objects. Such a world can be represented as a set of rigid planes moving in 3D, $\mathcal{P} = \{\bar{\mathbf{n}}, \mathcal{X}\}$, with parameters representing the plane normal $\bar{\mathbf{n}}$ and motion $\mathcal{X}$. In the ideal case, a slanted plane projects back to one or more superpixels in the images, inside of which the appearance and geometry information are locally similar. The inverse problem is then to infer the 3D planes (parameters $\bar{\mathbf{n}}$ and $\mathcal{X}$), given the images and a set of pre-computed superpixels.

3D Plane: We denote a plane as $\bar{\mathbf{n}}$ in 3-space, specified by its normal coordinates in the reference frame. For any 3D point $\mathbf{x} \in \mathbf{R}^3$ on $\bar{\mathbf{n}}$, the plane equation holds as $\bar{\mathbf{n}}^\top \mathbf{x} + 1 = 0$. We choose this parameterization for ease of optimization on its manifold.

Plane Motion: A rigid plane transform $\mathcal{X} \in \mathbf{SE3}$ comprising rotation and translation is defined by

$$\mathcal{X} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}, \mathbf{R} \in \mathbf{SO3}, \mathbf{t} \in \mathbb{R}^3 \tag{4.1}$$

Superpixel Associations: We assume each superpixel S_i is a one-to-one mapping in reference frame to a 3D plane. The boundary between adjacent superpixels S_i and S_j is defined as $\mathcal{E}_{i,j} \in \mathbb{R}^2$.

Transformation induced by moving planes: For any point $\mathbf{x}$ on $\bar{\mathbf{n}}$, its homogeneous representation is $[\mathbf{x}^\top, -\bar{\mathbf{n}}^\top \mathbf{x}]$. From $\mathbf{x}_0$ in the reference frame, its corresponding point $\mathbf{x}_1$ in an observed frame is:

$$\begin{bmatrix} \mathbf{x}_1 \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_0^1 & \mathbf{t}_0^1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_0 \\ -\bar{\mathbf{n}}^T \mathbf{x}_0 \end{bmatrix} \tag{4.2}$$

where $[\mathbf{R}_0^1 | \mathbf{t}_0^1]$ is the transform from reference frame to the observed image frame (referred to as $\mathcal{T}_0^1$) and $[\mathbf{R}_i | \mathbf{t}_i]$ is the plane motion in the reference frame (referred to as $\mathcal{X}_i$). Suppose the camera intrinsic matrix is $\mathbf{K}$. A homography transform can thus be induced as:

$$\begin{aligned} \mathbf{H}(\mathcal{P}_i, \mathcal{T}_0^1) &= \mathbf{K}[\mathbf{A} - \mathbf{a}\bar{\mathbf{n}}]\mathbf{K}^{-1} \\ \begin{bmatrix} \mathbf{A} & a \\ 0 & 1 \end{bmatrix} &= \begin{bmatrix} \mathbf{R}_0^1 & \mathbf{t}_0^1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{4.3}$$

In stereo frames where planes are static, the homography from reference frame to the right frame is simply:

$$\mathbf{H}(\bar{\mathbf{n}}, \mathcal{T}_{r \to 0}) = \mathbf{K}(\mathbf{R}_0^r - \mathbf{t}_0^r \bar{\mathbf{n}})\mathbf{K}^{-1} \tag{4.4}$$

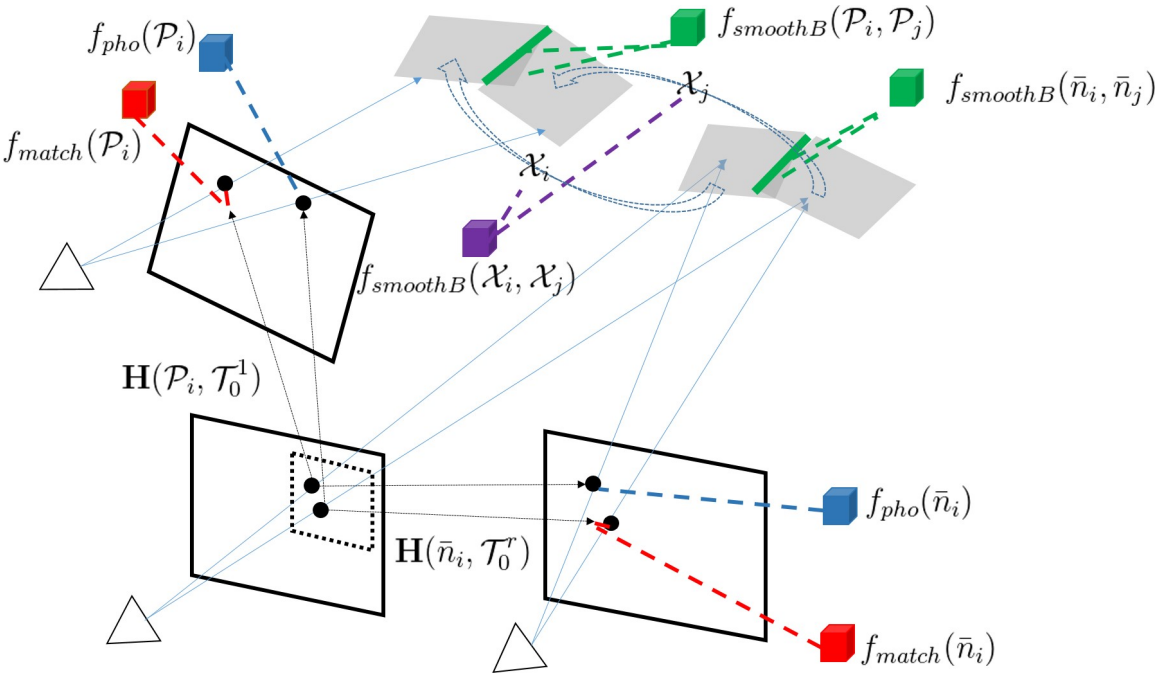


Figure 4.1: **A factor graph representation of multi-view constraints of stereo scene flow**. The unary factors are set up based on the homography transform relating two pixels, given $\mathcal{P}$. Binary factors are set up based on locally smooth and rigid assumptions. In this graph, a three-view geometry is used to explain factors for simplicity. Any other views can be constrained by incorporating the same temporal factors in this graph.

We will only use $\mathcal{T}_0^r$ to represent the transform of reference frame to the other stereo frame, while $\mathcal{T}_0^1$ is applicable from reference frame to any other frames, whether planes are static or moving.

4.3 A Factor Graph Formulation for Scene Flow

For all images $I' : \Omega \rightarrow \mathbb{R}$ relative to the reference image $I : \Omega \rightarrow \mathbb{R}$, we want to estimate all of the planes $\Theta = \{\bar{\mathbf{n}}_{\{1...N\}}, \mathcal{X}_{\{1...N\}}\}$ observed in I. Besides raw image measurements, we also assume that a set of sparsely matched point pairs $M \in \mathbb{R}^2$ is available. As mentioned above, we assume an a-priori fixed superpixel segmentation S, along with its boundaries $\mathcal{E}$. We denote these as our measurements $\mathcal{M} = \{I, I', M, S, \mathcal{E}\}$.

We begin by defining parameters $\theta = \{\bar{\mathbf{n}}, \mathcal{X}\}$, in which $\bar{\mathbf{n}}$ and $\mathcal{X}$ are independent to each other. We also assume dependencies only exist between superpixels across common

edges. The joint probability distribution of Θ can then be

$$\mathbf{P}(\Theta, \mathcal{M}) \propto \prod_{i \in N} \mathbf{P}(\theta_i|\mathcal{M}) \prod_{j \in N \setminus \{i\}} \mathbf{P}(\theta_i, \theta_j|\mathcal{M}) \tag{4.5}$$

$$\mathbf{P}(\theta_i|\mathcal{M}) \propto \mathbf{P}(I', M|\bar{\mathbf{n}}_i, \mathcal{X}_i, S_i, I)\mathbf{P}(\bar{\mathbf{n}}_i)\mathbf{P}(\mathcal{X}_i) \tag{4.6}$$

$$\mathbf{P}(\theta_i, \theta_j|\mathcal{M}) = \mathbf{P}(\bar{\mathbf{n}}_i, \bar{\mathbf{n}}_j|S_i, S_j, \mathcal{E}_{i,j})\mathbf{P}(\mathcal{X}_i, \mathcal{X}_j|S_i, S_j, \mathcal{E}_{i,j}), \tag{4.7}$$

Factor graphs are convenient probabilistic graphical models for formulating the multi-view constraints in scene flow problem

$$G(\Theta) = \prod_{i \in N} f_i(\theta_i) \prod_{i,j \in N} f_{ij}(\theta_i, \theta_j), \tag{4.8}$$

Typically $f(\theta_i)$ encodes a prior or a single measurement constraint at unknown θ, and $f_{i,j}$ relate to measurements or constraints between θ_i, θ_j. In this paper, we assume each factor is a least-square error term with Gaussian noises. To fully represent the measurements and constraints in this problem, we will use multiple factors for $G(\Theta)$ (see Fig. 4.1), which will be illustrated below.

Unary Factors: A point p, associated with a particular superpixels, can be associated with the homography transform $\mathbf{H}(\bar{\mathbf{n}}_i, \mathcal{C}_s)$ w.r.t. its measurements. For a stereo camera, the transformation of a point from one image to the other is simply $\mathbf{H}(\bar{\mathbf{n}}, \mathcal{C}_s)$ in (4.4). For all the pixels p in superpixel S_i, their photometric costs given $\mathcal{P}\{\bar{\mathbf{n}}_i, \mathcal{X}_i\}$ is

$$f_{pho}(\mathcal{P}_i) \propto \prod_{p \in S_i} f\big(C(p'), C(\mathbf{H}(\mathcal{P}_i, \mathcal{T}_{1\to 0})p\big), \tag{4.9}$$

where $C(\cdot)$ is the Census descriptor. This descriptor is preferred over intensity error for its robustness against noise and edges. Similarly, using the homography transform and with sparse matches we can estimate the geometric error of match m by measuring its

consistency with the corresponding plane motion

$$f_{match}(\mathcal{P}_i) \propto \prod_{p \in S_i} f\big(p + m, \mathbf{H}(\mathcal{P}_i, \mathcal{T}_0^1)p\big), \tag{4.10}$$

Pairwise Factors: The pairwise factors relate the parameters based on their constraints. $f_{smoothB}(\cdot, \cdot)$ describes the locally smooth assumption that adjacent planes should share similar boundary connectivity

$$f_{smoothB}(\bar{\mathbf{n}}_i, \bar{\mathbf{n}}_j) \propto \prod_{p \in \mathcal{E}_{i,j}} f\big(D^{-1}(\bar{\mathbf{n}}_i, p), D^{-1}(\bar{\mathbf{n}}_j, p)\big), \tag{4.11}$$

where $D^{-1}(\bar{\mathbf{n}}, p)$ represents the inverse depth of pixel p on $\bar{\mathbf{n}}$. This factor describes the distance of points over the boundary of two static planes. After plane motion, we expect the boundary to still be connected after the transformation:

$$f_{smoothB}(\mathcal{P}_i, \mathcal{P}_j) \propto \prod_{p \in \mathcal{E}_{i,j}} f\big(D^{-1}(\mathcal{P}_i, p), D^{-1}(\mathcal{P}_j, p)\big), \tag{4.12}$$

We also expect that two adjacent superpixels has piece-wise smooth motion. We use a *Between* operator of $\mathbf{SE3}$, described by $f_{smoothM}$ as

$$f_{smoothM}(\mathcal{X}_i, \mathcal{X}_j) \propto f\big(\mathcal{X}_i, \mathcal{X}_j\big). \tag{4.13}$$

Gaussian Noise Factor: Each factor is created as a Gaussian Noise Model:

$$F(\mathbf{x}; m) = \exp(-\rho(h(\mathbf{x}) - \mathbf{m})) \tag{4.14}$$

in which $\mathbf{m}$ is the measurement, and $h(\mathbf{x})$ is the estimation function of $\mathbf{x}$. We use the robust loss function $\rho(\mathbf{e})$ to handle the effect of outliers. For all the factors mentioned in the paper, we use the Huber kernel as loss function defined in Section 3.2.3.

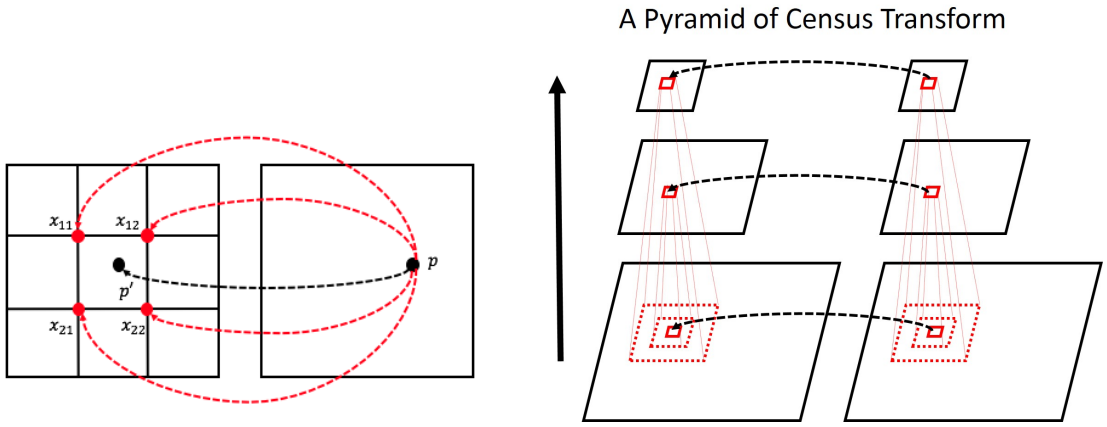


Figure 4.2: **Continuous Approximation of Census Transform.** The left figure shows how to use bilinear interpolation to achieve a differentiable cost of Census Transform. In the right, a census descriptor is extracted at different pyramid levels of the images.

Least-square Optimization: The factor graph in (4.5) can be estimated via maximum a posteriori (MAP) as a non-linear least square problem, and solved with standard non-linear optimization methods. In each step, we linearize all the factors at $\theta = \{\bar{\mathbf{n}}_\theta, \mathcal{X}_\theta\}$. On manifold, the update is a *Retraction* $\mathcal{R}_\theta$. The retraction for $\{\bar{\mathbf{n}}, \mathcal{X}\}$ is:

$$\mathcal{R}_\theta(\delta\bar{\mathbf{n}}, \delta\mathcal{X}) = (\bar{\mathbf{n}} + \delta\bar{\mathbf{n}}, \mathcal{X}\text{Exp}(\delta x)), [\delta\bar{\mathbf{n}} \in \mathbb{R}^3, \delta x \in \mathbb{R}^6] \tag{4.15}$$

For $\bar{\mathbf{n}} \in \mathbb{R}^3$, it has the same value of its tangent space at any value $\hat{n}$. This explains our choice of plane representation: it is the most convenient for manifold optimization in all of its families in 3-space. For $\mathbf{SE3}$ motion, the retraction is an exponential map. Although the linearized factor graph can be thought of as a huge matrix, it is actually quite sparse in nature: pairwise factors only exist between adjacent superpixels. Sparse matrix factorization can solve this kind of problem very efficiently. We follow the same sparse matrix factorization which is discussed in detail in [80].

Continuous Approximation for Census Transform: In (4.9), there are two practical issues: first, we cannot get a sub-pixel Census Transform; and second, the Hamming distance between the two descriptors is not differentiable. To overcome these problems, we use bi-

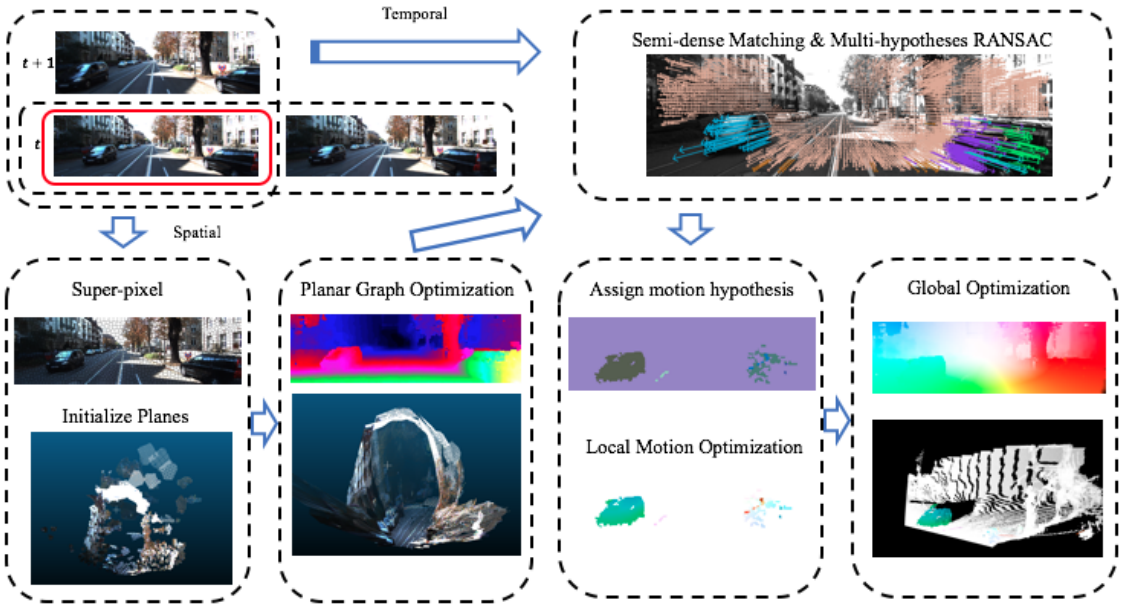


Figure 4.3: **An overview of the proposed continuous scene flow optimization method**: we estimate the 3D scene flow w.r.t. the reference image (the red bounding box), a stereo image pair and a temporal image pair as input. Image annotations show the results at each step. We assign a motion hypothesis to each superpixel as an initialization and optimize the factor graph for more accurate 3D motion. Finally, after global optimization, we show a projected 2D flow map in the reference frame and its 3D scene motion (static background are plotted in white).

linear interpolated distance as the census cost (see Fig. 4.2). The bilinear interpolation equation is differentiable w.r.t. the image coordinate, from which we can approximately get the Jacobian of Census Distance w.r.t. to a sub-pixel point. We use a 9x7 size Census, and set up (4.9) over a pyramid of images. In evaluation, we will discuss how this process helps us to achieve better convergence purely with a data-cost.

4.4 Scene Flow Optimization Pipeline

The general pipeline of our algorithms consists of five steps (see Fig. 4.3). We summarize each step and provide detailed descriptions in the subsections below.

Initialization: We initialize the superpixels for the reference frame. We estimate the 3D plane using an initial depth map using RANSAC.

Planar Graph Optimization: We solve the plane geometry parameter $\bar{\mathbf{n}}$ w.r.t. reference frame using factor graph composed of factors in (4.9), (4.10) and (4.11).

Estimation of Motion Hypotheses: We first estimate a semi-dense matching from reference frame to the next temporal frame and associate them with our estimated 3D plane to get a set of 3D features. We use RANSAC to heuristically find a set of motion hypothesis. In each RANSAC step, we find the most likely motion hypothesis of (4.3) by minimizing the re-projection errors of 3D features in two temporally consecutive frames. A set of motion hypotheses are generated by iterating this process.

Local Motion Graph Optimization: We initialize the motion of superpixels from the set of motion hypotheses, framed as a Bayesian classification problem. For all of the superpixels assigned to one single motion hypothesis, we estimate both the plane $\bar{\mathbf{n}}$ and its motion $\mathcal{X}$, by incorporating factors in (4.9), (4.12), (4.13).

Global Graph Optimization: In this step, the set of all unknowns $\mathcal{P}$ is estimated globally. All factors from (4.9) to (4.13) are used.

4.4.1 Initialization

The superpixels in the reference frame are initialized with the sticky-edge superpixels introduced in [82]. Since the urban scene is complex in appearance, the initialized superpixel number needs to be large to cope with tiny objects, while too many superpixels can cause an under-constrained condition for some plane parameters. Empirically, we find generating 2,000 superpixels is a good balance (see ablation in Table 4.3)

We use the stereo method proposed in [83] to generate the stereo prior, and initialize the 3D planes with a plane-fitting RANSAC algorithm. The plane is initialized as frontal parallel if the RANSAC inlier percentage is below a certain threshold (50% in our setting), or the plane induces a degenerated homography transform (where the plane is parallel to the camera focal axis).

We sample robust matches $\mathcal{M}$ from the disparity map, and use it to set up the matching factor in (4.10). The samples are selected from the Census Transform which share a maximum distance of 3 bits, given the disparity matching.

4.4.2 Planar Graph Optimization

In the stereo factor graph, we only estimate the planes $\bar{\mathbf{n}}$ from the factors in (4.9), i.e. we constrain the motion $\mathcal{X}$ to be constant. Suppose for each Gaussian noise factor, r is its residual: $f(x) = \exp(-r(x))$. We can obtain the maximum a posterior (MAP) of the factor graph by minimizing the residuals in the least-square problem:

$$\bar{\mathbf{n}}^{\star} = \underset{\bar{\mathbf{n}}}{\operatorname{argmax}} \prod f_{pho}(\bar{\mathbf{n}}_i) \cdot \prod f_{match}(\bar{\mathbf{n}}_i) \cdot \prod f_{smoothB}(\bar{\mathbf{n}}_i, \bar{\mathbf{n}}_j) \tag{4.16}$$

$$= \underset{\bar{\mathbf{n}}}{\operatorname{argmin}} \sum r_{pho}(\bar{\mathbf{n}}_i) + \sum r_{match}(\bar{\mathbf{n}}_i) + \sum r_{smoothB}(\bar{\mathbf{n}}_i, \bar{\mathbf{n}}_j) \tag{4.17}$$

Levenberg-Marquardt can be used to solve this equation as a more robust choice (e.g. compared to Gauss-Newton), trading off efficiency for accuracy.

4.4.3 Semi-dense Matching & Multi-Hypotheses RANSAC

We leverage the state-of-art matching method [18] to generate a semi-dense matching field, which has the advantage of being able to associate across large displacements in the image space. To estimate the initial motion for superpixels, we chose RANSAC similar to [13]. We classify putatives as inliers based on their re-projection errors. The standard-deviation $\sigma = 1$ is small to ensure that bad hypotheses are rare. All hypotheses with more than 20% inliers in each step are retained. Compared to the up-to-5 hypotheses in [13], we found empirically that our RANSAC strategy can retrieve 10-20 hypotheses in complex scenes, which ensures a high recall of even small moving objects, or motion patterns on non-rigid objects (e.g. pedestrians and cyclists). This process can be quite slow when noisy matches are prominent and inliers ratios are low. To cope with this effect, we use

Figure 4.4: A visualization of motion hypothesis (left), optical flow (middle), and scene motion flow (right). Camera motion is explicitly removed from scene motion flow. In the image of the cyclist we show that although multiple motion hypotheses are discovered by RANSAC (in two colors), our final continuous optimization can obtain a smooth motion over this non-rigid entity.

superpixels as a prior in RANSAC. We evaluate the inlier superpixels (indicated by inlier feature matches through non-maximum suppression), and reject conflicting feature matches as outliers. This prunes the number of motion hypotheses, and substantially speeds up this step. See Figure 4.4 for an illustration of the motion hypotheses.

Since the most dominant transform in the scene is induced by the camera transform, we can get an estimate of the incremental camera transform in the first iteration. After each iteration, the hypothesis is refined by a weighted least squares optimization, solved efficiently by Levenberg-Marquardt.

4.4.4 Local Motion Estimation

After estimation of the plane itself, we initialize the motion $\mathcal{X}_i$ of each individual plane from the set of motion hypotheses. At this step, given the raw image measurements $I_{0,1}$, a pair of estimated depth maps in both frames $D_{0,1}$, and the sparse point-matching field F, the goal is to estimate the most probable hypothesis $l^\star$ for each individual superpixel. We assume a set of conditional independencies among $I_{0,1}$, $D_{0,1}$, and F, given the superpixel.

The label l for each superpixel can therefore be inferred from the Bayes rule:

$$P(l|F, I_{0,1}, D_{0,1}) \propto P(F, I_{0,1}, D_{0,1}|l)P(l) \tag{4.18}$$

$$\propto P(I_{0,1}|l)P(D_{0,1}|l)P(F, I_0, D_0|l)P(l), \tag{4.19}$$

Assuming each motion hypothesis has equally prior information, a corresponding MAP estimation to the above equation can be presented as:

$$l^\star = \underset{l^\star}{\operatorname{argmax}}\, \mathbf{E}_{depth}(l) + \alpha \mathbf{E}_{photometric}(l) + \beta \mathbf{E}_{cluster}(l), \tag{4.20}$$

where $\mathbf{E}_{depth}(l)$ represents the depth error between the warped depth and transformed depth, given a superpixel and its plane; $\mathbf{E}_{photometric}(l)$ represents the photometric error between the superpixel and its warped superpixel; $\mathbf{E}_{cluster}(l)$ represents the clustering error of a superpixel, w.r.t. its neighborhood features:

$$\mathbf{E}_{depth}(l) = \sum_{p_i \in S} (D_1(\mathbf{H}p_i)) - z(\mathbf{H}p_i))^2, \tag{4.21}$$

$$\mathbf{E}_{photometric}(l) = \sum_{p_i \in S} (I(p_i) - I(\mathbf{H}p_i))^2, \tag{4.22}$$

$$\mathbf{E}_{cluster}(l) = \sum_{p_i \in S} \sum_{p_k \in F_l} \exp(-\frac{\bigtriangledown I_{i,k}^2}{\sigma_i^2}) \exp(-\frac{\bigtriangledown D_{i,k}^2}{\sigma_D^2}), \tag{4.23}$$

where $\mathbf{H}$ is the homography transform and $z(p)$ is the depth at pixel p. $\bigtriangledown I_{i,k}^2$ and $\bigtriangledown D_{i,k}^2$ describes the color and depth difference of a pixel $p_i \in S$ to a feature point $p_k \in F_l$ belonging to hypothesis l. σ_I and σ_D are their variances.

A local motion optimization is done for each hypothesis by incorporating the factors

(4.9), (4.10), (4.12), (4.13) with pre-estimated planes values as:

$$\mathcal{X}^{\star} = \underset{\mathcal{X}}{\operatorname{argmin}} \sum r_{pho}(\mathcal{X}_i) + \sum r_{match}(\mathcal{X}_i) + \sum r_{smoothB}(\mathcal{X}_i, \mathcal{X}_j) + \tag{4.24}$$

$$\sum r_{smoothM}(\mathcal{X}_i, \mathcal{X}_j) + \sum r_{prior}(\mathcal{M}). \tag{4.25}$$

Similar to (4.16), r is the residual for each factor. We add a prior factor $f_{prior}(\cdot)$ to enforce an L_2 prior centered at 0 It works as a diagonal term to improve the condition numbers in the matrix factorization. The prior factor has small weights and in general do not affect the accuracy or speed significantly.

4.4.5 Global Optimization

Finally, we estimate the global factor graph, with the complete set of parameters $\mathcal{P} = \{\bar{\mathbf{n}}, \mathcal{X}\}$ in the reference frame. The factors in this stage are set using measurements in all of the other three views, w.r.t. reference image.

$$\mathcal{P}^{\star} = \underset{\mathcal{P}}{\operatorname{argmin}} \sum r_{pho}(\mathcal{P}_i) + \sum r_{match}(\mathcal{P}_i) + \sum r_{smoothB}(\mathcal{P}_i, \mathcal{P}_j) + \tag{4.26}$$

$$\sum r_{smoothM}(\mathcal{P}_i, \mathcal{P}_j) + \sum r_{prior}(\mathcal{P}_i) \tag{4.27}$$

4.5 Experiments

4.5.1 Quantitative Results

KITTI Scene Flow Results: We evaluate our algorithm on the challenging KITTI Scene Flow benchmark [13], which is a realistic benchmark in outdoor environments. At the time of submission, our method ranks *3rd in Scene Flow test* while being significantly faster than close competitors, as well as *3nd in the KITTI Optical Flow test* and 11th in the stereo test which we did not explicitly target.

Table 4.1 shows a comparison of our results to other state-of-art optimization based

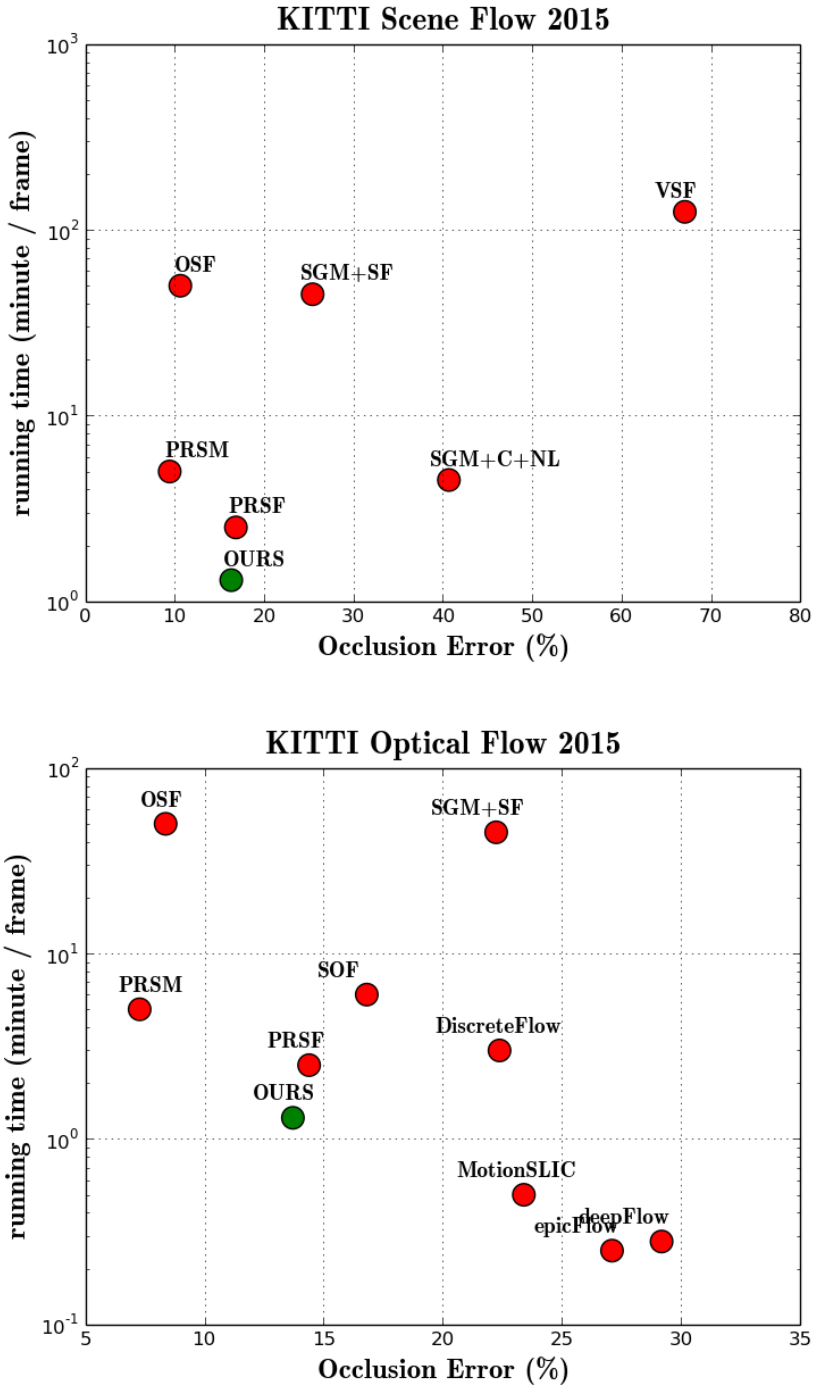


Figure 4.5: **Occlusion error-vs-time on KITTI**. The running time axis is plotted in log scale. Our method is highlighted as green, which achieves top performance both in accuracy and computation speed.

Table 4.1: **Quantitative Results on KITTI Scene Flow Test Benchmark** at the time of submission (March 20, 2016). We show the disparity errors reference frame (D1) and second frame (D2), flow error (Fl), and the scene flow (SF) in 200 test images on KITTI. The errors are reported as background (bg), foregound (fg), and all pixels (bg+fg), OCC for errors over all areas, NOC only for errors non-occluded areas.

Method	Occlusion (OCC) error												
	D1			D2			Fl			SF			
	bg%	fg%	all%	bg%	fg%	all%	bg%	fg%	all%	bg%	fg%	all%	time
PRSM[84]	**3.02**	**10.52**	**4.27**	**5.13**	**15.11**	**6.79**	**5.33**	**17.02**	**7.28**	**6.61**	**23.60**	**9.44**	300 s
OSF [13]	4.54	12.03	5.79	5.45	19.41	7.77	5.62	22.17	8.37	7.01	28.76	10.63	50 min
PRSF [15]	4.74	13.74	6.24	11.14	20.47	12.69	11.73	27.73	14.39	13.49	33.72	16.85	150 s
SGM+SF [85]	5.15	15.29	6.84	14.10	23.13	15.60	20.91	28.90	22.24	23.09	37.12	25.43	45 min
VSF [17]	27.73	21.72	26.38	59.51	44.93	57.08	50.06	47.57	49.64	67.69	64.03	67.08	125 min
Ours	4.57	13.04	5.98	7.92	20.76	10.06	10.40	30.33	13.71	12.21	36.97	16.33	**80 s**

Method	Non-Occlusion (NOC) error												
	D1			D2			Fl			SF			
	bg%	fg%	all%	bg%	fg%	all%	bg%	fg%	all	bg%	fg%	all	time
PRSM[84]	**2.93**	**10.00**	**4.10**	**4.13**	**12.85**	**5.69**	4.33	**14.15**	**6.11**	**5.54**	**20.16**	**8.16**	300 s
OSF [13]	4.14	11.12	5.29	4.49	16.33	6.61	**4.21**	18.65	6.83	5.52	24.58	8.93	50 min
PRSF [15]	4.41	13.09	5.84	6.35	16.12	8.10	6.94	23.64	9.97	8.35	28.45	11.95	150 s
SGM+SF [85]	4.75	14.22	6.31	8.34	18.71	10.20	13.36	25.21	15.51	15.28	32.33	18.33	45 min
VSF [17]	26.38	19.88	25.31	52.30	40.83	50.24	41.15	44.16	41.70	61.14	60.38	61.00	125 min
Ours	**4.03**	11.82	5.32	6.39	16.75	8.25	8.72	26.98	12.03	10.26	32.58	14.26	**80 s**

approaches. In all of these results, the errors in disparity and flow evaluation are counted if the disparity or flow estimation exceeds 3 pixels and 5% of its true value. In the Scene Flow evaluation, the error is counted if any pixel in any of the three estimates (two stereo frame disparity images and flow image) exceed the criterion.

Accuracy v.s. Speed: We plot a error-vs-time figure in Fig. 4.5, which shows that our method achieves state-of-art performance, when considering both efficiency and accuracy.

Our results show a small difference in occlusion-errors, although occlusion is not directly handled as discrete labels. We follow the same representation in [15] and achieved

Table 4.2: **Quantitative Results on KITTI Optical Flow 2015 Dataset**. The errors are reported as background error(Fl-bg), foreground error (Fl-fg), and all pixels (Fl-bg+Fl-fg), NOC for non-occluded areas error and OCC for errors over all pixels. Methods that use stereo information are shown as *italic*.

Method	OCC error			NOC error			
	Fl-bg%	Fl-fg%	all%	Fl-bg%	Fl-fg%	all%	time
PRSM[84]	**5.33**	**17.02**	**7.28**	4.33	**14.15**	**6.11**	300 s
OSF[13]	5.62	22.17	8.37	**4.21**	18.65	6.83	50 min
PRSF[15]	11.73	27.32	14.39	6.94	23.64	9.97	150s
SOF [86]	14.63	27.73	16.81	8.11	23.28	10.86	6 min
SGM SF[85]	20.91	28.90	22.24	13.36	25.21	15.51	45 min
DiscreteFlow[87]	21.53	26.68	22.38	9.96	22.17	12.18	3 min
MotionSLIC [83]	14.86	66.21	23.40	6.19	64.82	16.83	30s
epicFlow [88]	25.81	33.56	27.10	15.00	29.39	17.61	**15s**
deepFlow [18]	27.96	35.28	29.18	16.47	31.25	19.15	17s
Ours	10.40	30.33	13.71	8.72	26.98	12.03	80 s

better performance in overall pixel errors and faster inference. Compared to all of these methods, our method is the fastest.

KITTI Optical Flow Results: Table 4.2 shows our method compared to state-of-art optical flow methods. Methods using stereo information are shown in italic. The deepFlow [18] and epicFlow [88] methods are also presented; these also leverage DeepMatching for data-association. Our method is third best for all-pixels estimation.

4.5.2 Ablation Study

The effects of choosing super-pixels: We show a visualization of superpixels in Fig. 4.7. In Table 4.3, we evaluate the number of super-pixels as priors and their effects over the final results. The evaluation are separately presented in first frame stereo error and the temporal frame flow error. The superpixel number shows the seeds we generate for superpixels,

which is not equal to the final number of superpixels. We merge all the superpixels smaller than a set of 3 pixels (impossible to generate a plane from this set) to their neighborhood.

From both stereo and flow results, we can see the optimal superpixel number is from 2000 to 4000, which delivers us the best estimation. Although smaller superpixels can better fit into the locally smooth and locally rigid assumption, we find the accuracy drops when too many superpixels are generated (8000 or more).

We also show the percentage of images that throw under-constraints exceptions when using Gauss-Newton Optimization methods. In general, these frames are solved by Levenberg-Marquart algorithm. The under-constraints issues can happen if the superpixel sizes are too small, or their measurements are treated as outliers. We find that it is inevitable in some frames in general, e.g. a tree leaf as a superpixel (it is an isolated superpixel both in appearance and geometry). However, too many under-constraint superpixels demonstrate that they worsen the system performance.

Choice of Factors: In Table 4.4, we evaluate the choice of each factor and their effects in the results. During motion estimation, we see that multi-scale Census has an important positive effect in improving convergence towards the optima. Note that the best choice of weights for each factor was tuned by using a similar analysis. A more detailed parameter analyses is presented in the supplement materials.

Contribution of Individual Steps: In Table 4.5, we generate intermediate results after each step and show their errors over the ground truth. An initialization step from RANSAC is crucial to the success of optimization. Skipping this step, the optimization will completely fail. These results were generated over the last 100 images in the KITTI training sets, which were used as validation for parameter tuning. In many scenes, optimizations of each sub-graph can be performed using Gauss Newton efficiently without losing much accuracy. However, there is a potential for under-constrained sub-problems to occur for some planes with limited measurements. We catch these exception when the system is ill-conditioned and use Levenberg-Marquart to solve the system instead. Otherwise, Gauss

Table 4.3: **Quantitative evaluation over superpixel sizes.** The errors are collected as the disparity errors reference frame (D1) and flow error (Fl) from the first 100 images on KITTI training data-set. The errors are reported as background (bg), foregound (fg), and all pixels (bg+fg) in non-occluded areas (Noc). In the column of Ill-conditioned frame, it shows the percentage of images in the total 100 frames that throw the an under-constraints exceptions when using Gauss-Newton Optimization methods. We highlight the row of 2000 superpixels, which is the final number we use as priors.

Stereo	D1 % (Noc)				
	D1-bg %	D1-fg %	D1-all %	time	Ill-conditioned frames %
100	9.32	23.03	12.35	10.0s	0
500	7.48	19.43	9.56	13 s	7
1000	4.97	12.60	5.85	15 s	11
2000	**4.17**	**10.28**	**4.89**	**22** s	**24**
4000	4.16	10.37	4.88	33 s	39
8000	4.40	11.02	5.05	58 s	96
Flow	F % (Noc)				
	F-bg %	F-fg %	F-all	time	Ill-conditioned frames %
100	23.0	50.61	32.32	20s	0
500	12.92	34.78	16.13	24s	8
1000	10.23	29.12	13.68	27s	13
2000	**8.53**	**26.23**	**12.00**	**31s**	**28**
4000	8.52	26.72	12.01	42s	45
8000	9.67	28.66	12.83	57s	96

Table 4.4: **Ablation of factors**. The non-occlusion error are used from 50 images of KITTI training set. The corresponding factors (in braces) are in Section 4.3.

Stereo	D1 % (Noc)		
	D1-bg %	D1-fg %	D1-all %
Census (4.9)	9.21	19.22	12.31
Matching (4.10)	5.95	15.20	7.62
Census + Matching (4.9, 4.10)	5.66	15.01	6.93
Census + Continuity (4.9, 4.11)	4.85	14.22	5.94
All (4.9, 4.10, 4.11)	4.13	10.20	4.85
Flow	F % (Noc)		
	F-bg %	F-fg %	F-all
Census Raw only (4.9)	10.9	34.25	14.20
Census Multi-scale (4.9)	9.3	30.13	12.45
Matching only (4.10)	10.5	33.40	13.20
Census+piecewise motion (4.9,4.13)	9.0	29.01	12.45
Census + continuity (4.9, 4.12)	9.2	30.15	12.44
All (4.9, 4.10, 4.13, 4.12)	8.92	28.92	12.31

Table 4.5: **Evaluation of errors of each step**(Non-occlusion score is reported from 100 images in training set of KITTI). The initialization in geometry estimation includes generating superpixels, stereo prior and plane-fitting ransac. The initialization in motion estimation includes DeepMatching and Multi-hypothesis RANSAC.

Stereo	D1 % (Noc)			
	D1-bg %	D1-fg %	D1-all %	time
Initialization	18.28	35.32	19.35	5.0s
Stereo Optimize (Gauss-Newton)	4.98	13.20	5.74	6 s
Stereo Optimize (Levenberg-Marquart)	4.13	10.20	4.85	20 s
Global Optimize (Levenberg-Marquart)	4.15	10.18	4.86	22 s
Flow	F % (Noc)			
	F-bg %	F-fg %	F-all	time
Initialization	10.0	33.61	13.12	28s
Local Optimize (Gauss-Newton)	8.92	28.92	12.31	5s
Local Optimize (Levenberg-Marquart)	8.81	28.12	12.23	18s
Global Optimize (Levenberg-Marquart)	8.56	26.61	12.01	22s

Newton is preferred. In global optimization, we prefer Levenberg-Marquart algorithm due to better handling of the non-linearity in the energy function.

4.6 Conclusion

Our proposed approach solves the scene flow problem in continuous domain, resulting in a high accuracy (3rd) on the KITTI Scene Flow benchmark at a large computational speedup. We show that faster inference is achievable by rethinking the solution as a non-linear least-square problem, cast within a factor graph formulation. We then develop a novel initialization method, leveraging a multi-scale differentiable Census-based cost and DeepMatching. Given this initialization, we individually optimize geometry (stereo) and motion (optical flow) and then perform a global refinement using Levenberg-Marquardt. Analysis shows the positive effects of each of these contributions, ultimately leading to a fast and accurate scene flow estimation.

Although the proposed method has already achieved significant speed and accuracy, the performance of our optimization approach is not close to real-time performance. There are

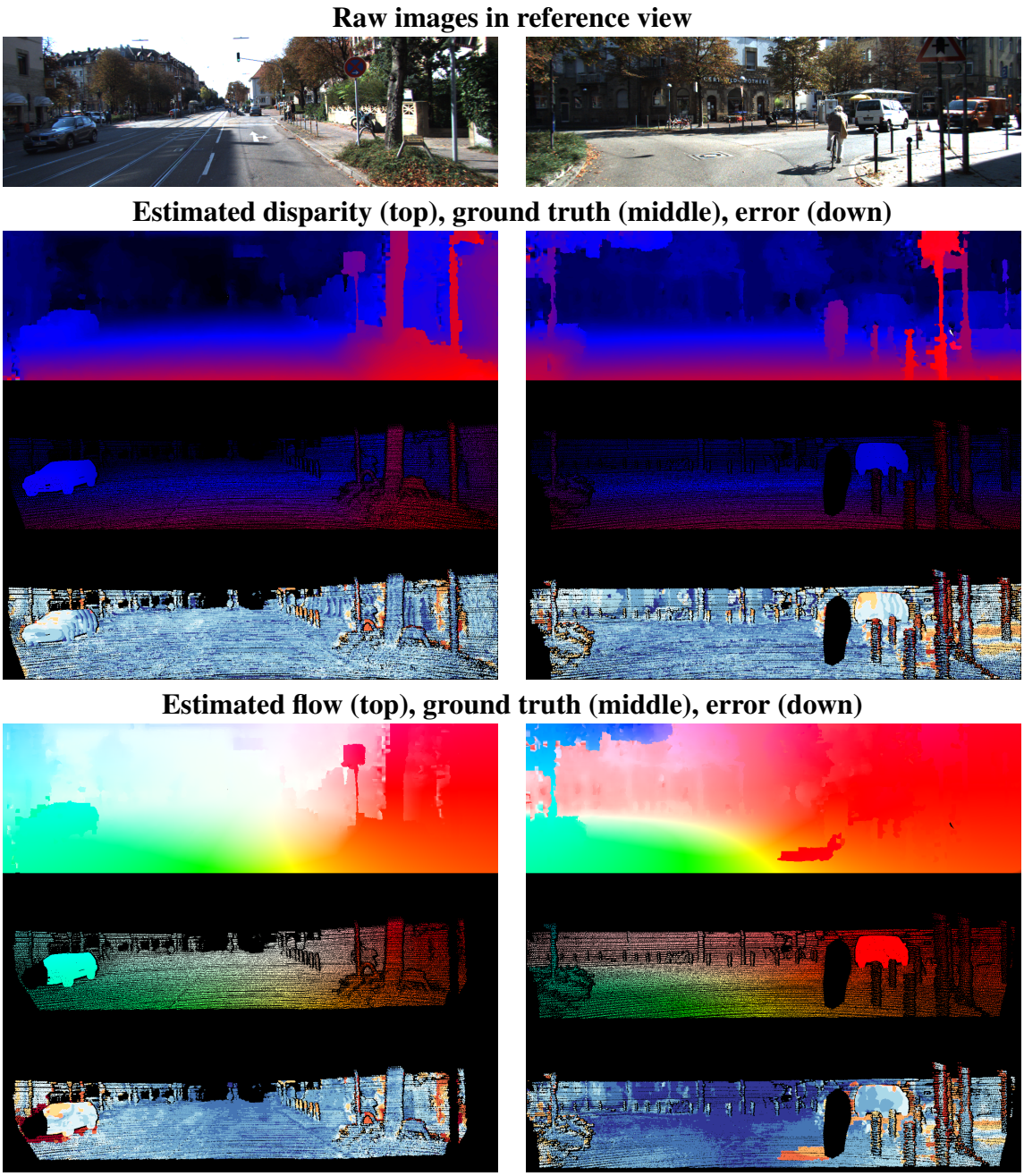
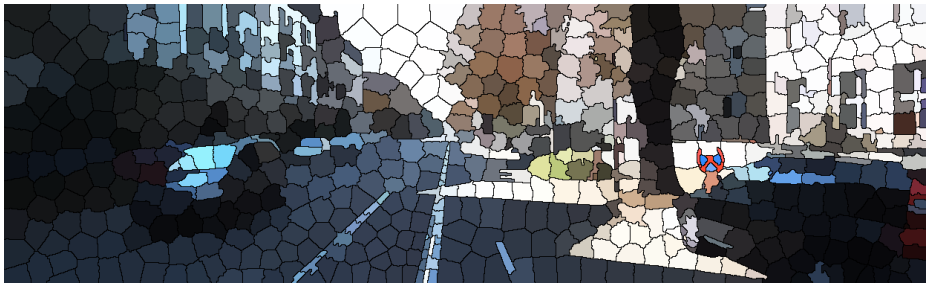


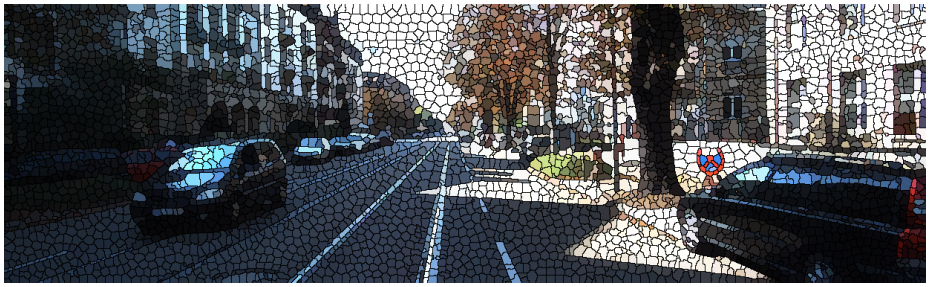
Figure 4.6: **Qualitative Results in KITTI.** We show the disparity and flow estimation against the ground truth results in Kitti Scene Flow training set.

(a) Raw Image



(b) 500 superpixels

(c) 2000 superpixels



(d) 8000 superpixels

Figure 4.7: A visualization of different size superpixels. When use small number of superpixels (500 or fewer), we lose the validity for local smooth and local rigid assumption. When an appropriate size superpixel is generated, we can visually see this assumption holds, even in tiny structured areas.

several challenging points and failure cases that we cannot resolve, such as photometric inconsistency in scenes and areas with aperture ambiguity. To address these problems, we expect to explore more invariant representations than the current unary factors, and more prior knowledge to enforce better local consistency.

CHAPTER 5

LEARNING SCENE FLOW ESTIMATION

Summary

Estimation of 3D motion in a dynamic scene from a temporal pair of images is a core task in many scene understanding problems. In real world applications, a dynamic scene is commonly captured by a moving camera (i.e., panning, tilting or hand-held), increasing the task complexity because the scene is observed from different view points. The main challenge is the disambiguation of the camera motion from scene motion, which becomes more difficult as the amount of rigidity observed decreases, even with successful estimation of 2D image correspondences. Compared to other state-of-the-art 3D scene flow estimation methods, in this paper we propose to *learn* the rigidity of a scene in a supervised manner from a large collection of dynamic scene data, and directly infer a rigidity mask from two sequential images with depths. With the learned network, we show how we can effectively estimate camera motion and projected scene flow using computed 2D optical flow and the inferred rigidity mask. For training and testing the rigidity network, we also provide a new semi-synthetic dynamic scene dataset (synthetic foreground objects with a real background) and an evaluation split that accounts for the percentage of observed non-rigid pixels. Through our evaluation we show the proposed framework outperforms current state-of-the-art scene flow estimation methods in challenging dynamic scenes.

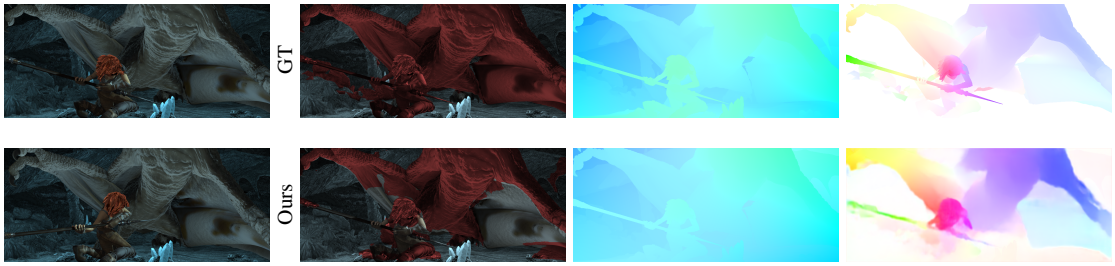
5.1 Introduction

Scene flow estimation from visual sequences in a dynamic environment is challenging when the scene is observed from different view points and the amount of coverage of moving objects in each image is significant. This is mainly because the disambiguation of camera motion (ego-motion) from object motion requires the correct identification of *rigid static structure* of a scene.

Recently there are remarkable progress in learning dense correspondences in 2D tasks, e.g. optical flow [19, 46, 48] and these methods have demonstrated the ability to can generalize to new scenes. The success of learning inspires us to view scene flow as a correspondence regression problem. If we can disentangle scene flow in 3D domain from the optical flow in 2D observations, we should also be able to learn scene flow.

To disambiguate optical flow induced by ego-motion from scene flow requires the correct identification of the static structure of a scene, which has been called *rigidity* [89]. Wulff et al. [89] assume there is a high correlation of rigidity to the semantic information, and thus we can use semantic segmentation as prior knowledge to isolate rigid motions. However, using semantic information as prior knowledge often does not stand true on its own. Many objects can move by external forces although most of the time they remain static. Solving rigidity in scene flow has also been tacked from the classical scene flow approaches using other assumptions, such as piecewise rigid motion [84, 90], clustering local motions [39], and semantic grouping [36]. Random Sample Consensus (RANSAC) and its variants are one of the most widely used techniques to extract the motion inliers based on the statistics but it can also often fail if used alone when the motions are complex and ambiguous.

Unlike previous work which defines the rigidity using intermediate information, we propose to train a network which takes two-views as input and can infer per-pixel rigidity purely from data. Our network jointly learns rigidity and the relative camera transform from



(a) Two RGB-D frames (b) Rigidity (red for dynamic scene) (c) Ego motion flow (d) Projected scene flow

Figure 5.1: Our estimated Rigidity (b), Ego-motion Flow (c) and Projected scene flow (d) (bottom row) compared to the ground truth (top row). The rigidity mask allows us to solve for the relative camera transform and compute the 3D motion field given the optical flow.

large-scale dynamic scene data. Our framework, shown in Fig. 5.2, takes a two sequential image pair as the input and mainly focuses on dynamic scenes with a moving camera (e.g., panning), where camera motion and objects motions are entangled in each observation. To solve for 2D correspondences, our framework relies on 2D optical flow, and is not tied to any particular algorithm. To provide better supervision during training and encourage generalization, we develop a tool and methodology that enables the creation of a scalable semi-synthetic RGB-D dynamic scene dataset, which we call *REFRESH*. This dataset combines real-world static rigid background with non-rigid synthetic human motions [91] and provides ground truth color, depth, rigidity, optical flow and camera pose.

Contributions: In summary, our major contributions are:

1. A learning-based rigidity and pose estimation algorithm for dynamic scenes with a moving camera.

2. An RGBD scene flow algorithm that builds on inference from rigidity, pose, and existing 2D optical flow, which outperforms the state-of-the-art methods. We have released our code repository is at https://github.com/NVlabs/learningrigidity.git.

3. A new semi-synthetic dynamic scene data and its creation tool: REal 3D From REconstruction with Synthetic Humans (REFRESH). We have made dataset and the entire rendering pipeline public at https://github.com/lvzhaoyang/RefRESH.git

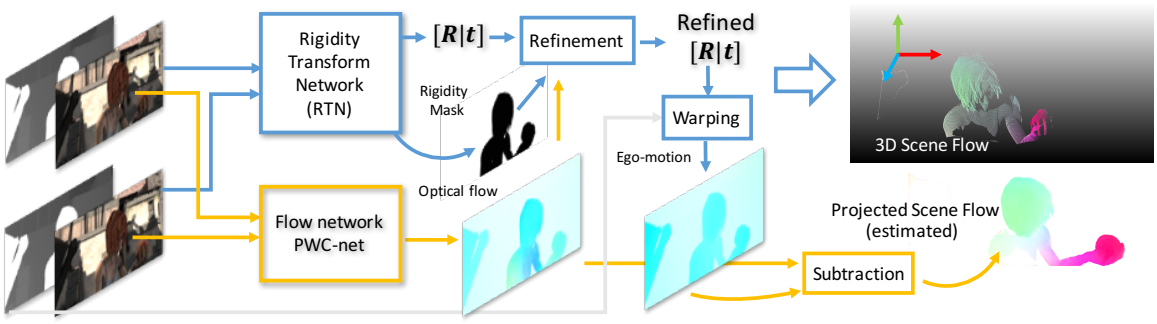


Figure 5.2: **An overview of our proposed inference architecture for 3D motion field estimation.** Our method takes two RGB-D frames as inputs independently processed by two networks. The Rigidity Transform Network (RTN) estimates the relative camera transform and rigid/non-rigid regions. The flow network [48] computes dense flow correspondences. We further refine the relative pose with dense flow over the rigid region. With the refined pose, we compute 3D motion field and projected scene flow from the ego-motion flow.

5.2 Two View Correspondences and Scene Flow

Here we define the relationship between 2D image correspondences and scene flow in physical 3D scenes with object motions and camera motion derived from relative camera poses between two temporal views.

Let $\mathbf{x}_t \in \mathbb{R}^3$ be the location of a point $\mathbf{x}$ on a non-rigid surface Ω_t of a moving object with respect to a fixed world coordinate system at time t. We define $\delta\mathbf{x}_{t\to t+1}$ as the 3D motion vector of $\mathbf{x}$ from time t to time $t+1$, which is scene flow discussed in this paper. When $\mathbf{x}_t$ is observed by a camera with known intrinsics, we define $\pi(\mathbf{x}_t)$ to be the projection of $\mathbf{x}_t$ to image coordinates $\mathbf{u}_t$, and $\pi^{-1}(\mathbf{u}_t, z_t)$ the inverse projection into 3D camera coordinates given the known depth z_t.

Scene flow, 2D Optical Flow, and Camera Pose: Optical flow offers direct 2D associations of measurements in I_t and I_{t+1}. Suppose $\mathcal{C}_t$ is a known camera extrinsic matrix from I_t. Then the optical flow $\delta\mathbf{u}_{t\to t+1}$ from I_t to I_{t+1} can be defined as follows:

$$\delta\mathbf{u}^{of}_{t\to t+1} = \pi(\mathcal{C}_{t+1}(\mathbf{x}_t + \delta\mathbf{x}_{t\to t+1})) - \pi(\mathcal{C}_t\mathbf{x}_t) \tag{5.1}$$

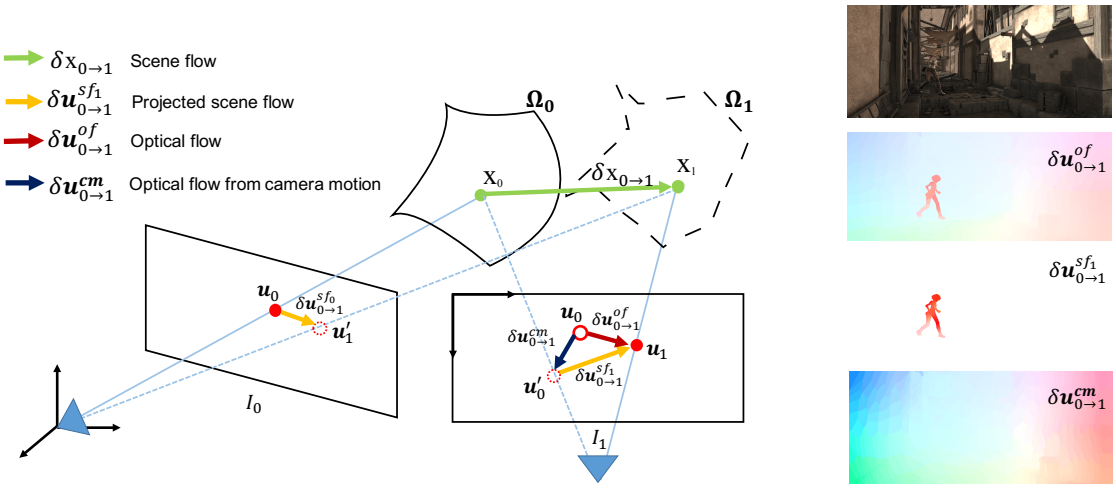


Figure 5.3: **Two-frame scene flow in a dynamic scene.** We show the geometry of a dynamic scene where the camera moves from I_0 to I_1, and point $\mathbf{x}_0$ moves to $\mathbf{x}_1$ (denoted as green circles), and their projections in the two images are shown as $\mathbf{u}_0, \mathbf{u}_1$ respectively (red circles). Note that $\mathbf{u}'_0$ is a projected location of $\mathbf{x}_0$ in I_1, as if $\mathbf{x_0}$ were observed by I_1, and can be computed by camera motion as $\delta \mathbf{u}^{cm}_{0\to1}$, and $\mathbf{u}_0$ in I_1 is visualizing the pixel location it had in I_0. If the camera was static and observed both $\mathbf{x_0}$ and $\mathbf{x_1}$ at the position of I_1, optical flow $\delta \mathbf{u}^{of}_{0\to1}$ would be same to a projected scene flow $\delta \mathbf{u}^{sf1}_{0\to1}$. The right image show each flow in I_1 of dynamic scene under camera panning.

(5.1) states the two-view geometric relationship between 2D optical flow and 3D scene flow. We can simplify it by considering the camera's relative motion from I_0 to I_1, i.e. assuming $t = 0$ and setting $\mathcal{C}_0$ to identity:

$$\delta \mathbf{u}^{of}_{0\to1} = \pi(\mathcal{C}_1(\mathbf{x}_0 + \delta \mathbf{x}_{0\to1})) - \pi(\mathbf{x}_0) \tag{5.2}$$

Given the optical flow $\delta \mathbf{u}^{of}_{0\to1}$ and the depth from the RGBD data, the 3D scene flow vector can be computed as:

$$\delta \mathbf{x}_{0\to1} = \mathcal{C}_1^{-1}\pi^{-1}(\mathbf{u}_0 + \delta \mathbf{u}^{of}_{0\to1}, z_1) - \pi^{-1}(\mathbf{u}_0, z_0) \tag{5.3}$$

Note that $\mathcal{C}_1$ can be computed from 2D correspondences that follow two-view epipolar geometry [92], and the corresponding points should lie on the rigid and static background structure. This is especially challenging when the scene contains dynamic components

(moving objects) as well as a rigid and stationary background structure. As such, identifying inliers and outliers using *rigidity* is a key element for successful relative camera pose estimation, and thus is necessary to achieve reaching accurate scene flow estimation in a dynamic scene [93, 39].

Egomotion Flow from a Moving Camera in a Static Scene: When an observed $\mathbf{x}$ in a scene remains static between the two frames, $\delta\mathbf{x}_{0\to1} = \mathbf{0}$ and therefore $\mathbf{x}_1 = \mathbf{x}_0$. Then, the observed optical flow is purely induced by the camera motion and we refer it as a camera egomotion flow:

$$\delta\mathbf{u}^{cm}_{0\to1} = \pi(\mathcal{C}_1\mathbf{x}_0) - \pi(\mathbf{x}_0) \tag{5.4}$$

Projected Scene Flow and Rigidity: As described in Fig. 5.3, the projected scene flow is a projection of a 3D scene flow $\delta\mathbf{x}_{0\to1}$ in I_1 if $\mathbf{x}_0$ was observed from I_1, which can be computed from camera ego-motion and optical flow:

$$\delta\mathbf{u}^{sf}_{0\to1} = \delta\mathbf{u}^{of}_{0\to1} - \delta\mathbf{u}^{cm}_{0\to1} \tag{5.5}$$

The projected scene flow (in a novel view) is also referred to non-rigid residual [38, 60]. All locations with zero values in projected scene flow indicate the rigidity region in ground truth data. As demonstrated in Fig. 5.3, the projected scene flow is a useful tool to evaluate the results of dense scene flow estimation in the 2D domain which requires accurate estimation of both camera pose and optical flow.

5.3 Learning Scene Flow Pipeline

Refer to the relationship between scene flow, optical flow and relative camera pose discussed in Section 5.2. we introduce a framework that refines the relative camera transform and the optical flow with a rigidity mask for accurate scene flow estimation.

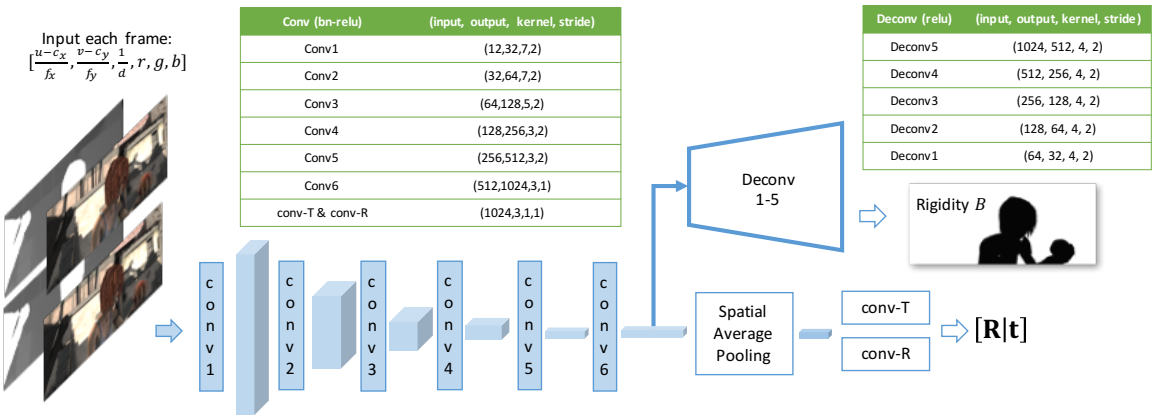


Figure 5.4: **Rigidity-Transform network (RTN) architecture** The inputs to the RTN are 12 channel tensors encoded with $[(u - c_x)/f_x, (v - c_y)/f_y, 1/d, r, g, b]$ computed from a pair of RGB-D images and camera intrinsics. The fully convolutional network predicts pose as a translation and euler angles, and scene rigidity as a binary mask.

Fig. 5.2 shows the overview of our proposed pipeline. Given a temporal pair of RGB-D images, we concurrently run the optical flow and rigidity-transform network. The flow network [48] offers the 2D correspondence association between frames, and our proposed rigidity-transform network provides an estimate of the camera transform and the rigidity mask. The camera pose is refined using the flow correspondences and the rigidity mask.

5.3.1 Rigidity-Transform Network

Previous work regressing pose focused on either purely static or quasi-static scenes, where scene motions are absent or their amount is minimal [54, 94, 57]. In dynamic scenes with a moving camera, camera pose estimation can be challenging due to the ambiguity induced by the camera motion and scene (object) motion. Existing solutions disambiguate the two using prior information in motion or semantic knowledge [39, 36, 38, 86]. We propose to replace this hand-coded criteria by a fully-convolutional network that jointly learns camera motion and a segmentation of the scene into dynamic and static regions. We represent this rigidity segmentation as a binary mask with the static scene masked as rigid. The rigid scene components will obey the epipolar constraints induced by the camera ego-motion and serve as the regions of *attention of the camera transform*. We name it rigidity-transform

network (RTN), shown in Fig. 5.4.

Given a pair of RGB-D frames, we pre-process each frame into a 6 channel tensor $[(u - c_x)/f_x, (v - c_y)/f_y, 1/d, r, g, b]$, from camera intrinsics parameters $[f_x, f_y, c_x, c_y]$ and the depth d. Considering the different range of depth values, this representation is numerical stable in training and delivers good generalization performance. We truncate $1/d$ to the range $[1e-4, 10]$, which is able to cover scenes of various scales. We concatenate the two-frame tensors to a 12-channel tensor as input to our network. The network is composed of an encoder followed by pose regression and a decoder followed by the rigidity.

Encoder: The encoder is composed of five stride-2 convolutional layers (1-5) which gradually reduce spatial resolution and one stride-1 convolution as the conv-6 layer. Each convolution is followed by a batchnorm and ReLU layer. The target is to predict the camera relative translation $\mathbf{t}$ and rotation Θ. After the conv-6 layer, we use a spatial-average pooling (SAP) to reduce the feature into a $1024D$ vector. With two 1×1 convolution layers that outputs 3 channels, we separately estimate the $\mathbf{t}$ and Θ. We assume the relative camera transformation between two frames is small and thus we represent the rotation $\mathbf{R}(\alpha, \beta, \gamma) = \mathbf{R}_x(\alpha)\mathbf{R}_y(\beta)\mathbf{R}_z(\gamma)$ with Euler angles $\Theta = [\alpha, \beta, \gamma]$. The regression loss is a weighted combination of the robust Huber loss $\rho(\cdot)$ for translation and rotation as:

$$\mathcal{L}_p = \rho(\mathbf{t} - \mathbf{t}^\star) + w_\Theta \rho(\Theta - \Theta^\star) \tag{5.6}$$

Decoder: The decoder network is composed of five deconvolution (transpose convolution) layers which gradually upsample the conv-6 feature into input image scale and reshape it into the original image resolution. We estimate the rigidity attention as a binary segmentation problem with binary cross-entropy loss $\mathcal{L}_r$. The overall loss is a weighted sum of both loss functions: $\mathcal{L}_c = w_p \mathcal{L}_p + \mathcal{L}_r$.

Enforcing Network Learning from Two Views: To learn the rigid regions of two views, we enforce the network to capture both scene structures and epipolar constraints w.r.t. two

views from two aspects. First, our network is fully convolutional and we regress the camera pose from the SAP layer which preserves feature distributions spatially. Features for rigidity segmentation and pose regression can interact directly with each other spatially across each feature map. We do not use any skip layer connections. Our experiments in Section **??** show simultaneously learning of camera pose and rigidity can help RTN achieve better generalization in complex scenes. Second, we randomly use two identical views as input and a fully rigid mask as output with 20% probability during data augmentation, which prevents the network from only using a single view for its prediction.

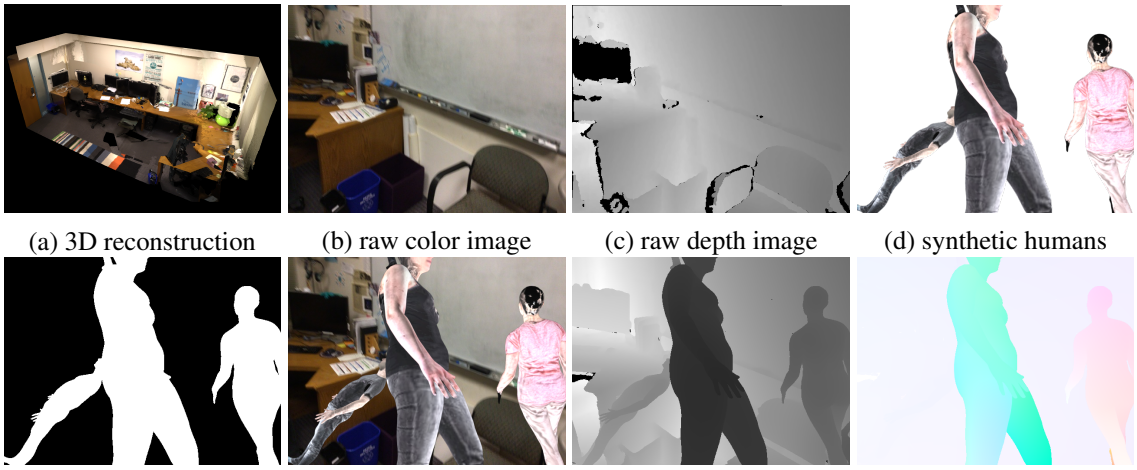
5.3.2 Camera Pose Refinement from Rigidity and Flow

From Section 5.2 we can compute the 3D motion field given the optical flow and the camera pose. To solve for the 3D motion field accurately from two views, we require a precise camera transformation. The pose output from RTN may not always precisely generalize to new test scenes. To overcome this, we propose a refinement step based on the estimated rigidity B and bidirectional dense optical flow $\delta\mathbf{u}^{of}_{0\to1}$ and $\delta\mathbf{u}^{of}_{1\to0}$ (with forward and backward pass). We view the estimation of $\mathcal{C}_1$ as a robust least square problem as:

$$\underset{\mathcal{C}_1}{\operatorname{argmin}} \sum_{\{\mathbf{x}_0,\mathbf{x}_1\}\in\Omega(B)} [\mathbf{I}]\rho(\mathcal{C}_1\mathbf{x}_0 - \mathbf{x}_1) \tag{5.7}$$

where $\mathbf{x}_i = \pi^{-1}(\mathbf{u}_i, z_i)$ in all background regions B, predicted by the RTN. $[\mathbf{I}]$ is an Iverson bracket for all the inlier correspondences.

The inlier correspondences are filtered in several steps. We first use forward backward consistency check for bidirectional optical flow with a threshold of 0.75 to remove all flow correspondences which are not consistent. This generates the occlusion map O. To prevent outliers at the boundary of rigidity B and occlusion O, we use a morphological operator with patch size 10 to dilate B and O. From all correspondences, we uniformly sample bidirectional flow correspondences with a stride of 4 and select 1e4 points among



(a) 3D reconstruction (b) raw color image (c) raw depth image (d) synthetic humans

(e) rendered rigidity (f) composited color (g) composited depth (h) composited gt flow

Figure 5.5: **REFRESH dataset creation pipeline** With a captured RGB-D trajectory, the scene is reconstructed as a 3D mesh by BundleFusion [95] (a), with raw RGB-D input as (b) and (c). With sampled frames from the camera trajectory, we load synthetic human models [91] with motions randomly into the 3D as (d), and render the rigidity mask (e), Finally we composite the rendered ground truth with its corresponding 3D views and the final semi-synthetic RGB-D views (f) and (h), with optical flow ground truth as (i).

them that are closest to the camera viewport. These help to solves the optimization more efficiently and numerically stable. We also use the Huber norm $\rho\cdot$ as a robust way to handle the remaining outliers.

We solve (5.7) efficiently via Gauss-Newton with $\mathcal{C}_1$ initialized from the RTN output. With accurate filtered correspondences, we found explicit initialize the pose using regression result is not necessary.

5.4 Create Training Dataset: REFRESH

Training our network requires a sufficient amount of dynamic RGB-D images in diverse scenes and ground truth in the form of known camera pose, rigidity mask, and optical flow. However, acquiring such ground truth from the real-world data is difficult or even infeasible. Existing dataset acquisition tools include rendered animations like SINTEL[96] and Monka[31], and frames captured from games [97]. SINTEL [96] has a small number of frames, so we use it for testing instead of training. Most approaches render scenes

using rigid 3D object models [98, 19, 31] with the concept. Among all existing tools and datasets, only Things3D[31] provides sufficient 3D training samples for learning 3D flow with moving camera ground truth. However, it only uses a small set of 3D objects with textured images at infinity as static scene context and rigid objects as the dynamic scene.

5.4.1 Dataset Rendering Details

To overcome the dataset issue, we propose a semi-synthetic scene flow dataset: REal 3D from REconstruction with Synthetic Humans, which we name as **REFRESH**. For this task we leverage the success of state of the art 3D reconstruction systems [95, 99, 100], which directly provide dense 3D meshes and optimized camera trajectories. We use a pre-captured RGB-D dataset and create dynamic 4D scenes by rendering non-rigid 3D moving objects with pre-defined trajectories. We overlay synthetic objects over the original footage to obtain a composite image with the ground truth as shown in Fig. 5.5.

We use Blender 2.78[1] to create the dataset, fully automated with python scripts without any GUI interaction, which scales well to the creation of the entire dataset. We separately render the background 3D meshes and foreground nonrigid humans, which allows us to speed up the rendering process. Since we use the raw color image as the background image and only use the geometry ground truth from multi-pass rendering (depth, flow, and segmentation), lighting does not affect background rendering with or without the foreground. Such separation can significantly boost the dataset creation speed. We can finish the entire rendering process using BundleFusion [95] 3D scenes in two days on a 28-core CPU server.

Real 3D Reconstructed Scenes: We use the 3D meshes created with BundleFusion [95]. The authors released eight reference 3D meshes with the 25K input RGB-D images, camera intrinsic and extrinsic parameters.

Synthetic humans: We create non-rigid scene elements with the method introduced in SURREAL [91] with synthetic textures (772 clothes textures and 158 CAESAR textures).

[1]Blender: https://www.blender.org/

The illuminated textures are used as the appearance of humans in our composted dynamic scenes. Each synthetic body is created from realistic articulated human body models [72] and pose actions are from the CMU MoCap database [101] with more than 20K sequences of 23 action categories. The human textures are composed of SMPL CAESAR scans and real clothing registered with Dyna [102]. We create each synthetic human with random gender, body shape, cloth texture, action and their positions in the 3D scene which guarantees the diversity of dynamic scenes. We control the visibility of human models along the trajectory by putting the pelvis point of each human model in the free space w.r.t. the ego-centric viewpoint from a selected frame along the trajectory. The free space is sampled by the corresponding depth. For every 100 frames, we select n frames (n sample from $\sim \mathcal{N}(15, 5)$) and insert n human models into the scene.

Rendering and ground-truth generation: We use Cycles from the Blender suite as our rendering engine. The lighting is created using spherical harmonics, as in [91]. First, we set the virtual camera using the same 3D scene camera intrinsic and spatial resolution. The camera extrinsic follows the real-data trajectory (computed from BundleFusion [95]). Thus, we can use the raw color image rather than rendered image as background texture which is photo-realistic and contains artifacts such as motion blur. With the same camera settings, we separately render the 3D reconstructed static mesh and the synthetic humans, and composite them using alpha-matting. Different from the color image, the depth map is rendered from the 3D mesh, which is less noisy and more complete than raw depth. Since the camera movement during the 3D acquisition is small between frames, we subsample frames at intervals of [1,2,5,10,20] to create larger motions. We employ a multi-pass rendering approach to generate depth, optical flow and rigidity mask as our ground truth.

We split the camera trajectory into multiple clips. Each clip is a continuous 100-frame sequence, with randomly loaded human models and actions. There are two major motivations to rendering the outputs in clips rather than an entire trajectory: 1. We can load different random human bodies and motions for different clips in the same trajectory, which

Table 5.1: The number of rendered images generated in our REFRESH dataset using BundleFusion [95] as 3D scenes.

	apt0	apt1	apt2	copyroom	office0	office1	office2	office3	Total
keyframe 1	8560	8495	3873	4478	6083	5727	3494	3757	44467
keyframe 2	4280	4248	1937	2239	3043	2863	1748	1882	22240
keyframe 5	1712	1700	776	895	1220	1146	700	752	8901
keyframe 10	856	849	338	447	609	572	349	376	4446
keyframe 20	427	424	195	223	304	286	174	189	2222
keyframe 50	171	169	78	89	123	114	69	75	888
Total	16006	15885	7247	8371	11382	10708	6534	5149	83164

increase the motion diversity both in action and appearance; 2. There are numerous human models generated along the entire trajectory, which composes complex meshes in 3D and slow for rendering. Rendering individual clip with several human models is much faster in execution, with an average of 3 seconds per frame.

We use the rendered depth from 3D scenes instead of the raw 3D scene depth for all the training. Compared to the raw depth, the rendered depth is less noisy and contains less missing measurements and has a per-pixel correspondence to the other ground truth, e.g., optical flow. However, the rendered depth does not guarantee a valid per-pixel value due to the incomplete 3D reconstruction from raw measurements. We marked the projected pixels from incomplete regions (holes in 3D reconstruction) as *invalid* region, and exclude them from the training on-the-fly.

5.4.2 Dataset statistics

We rendered dataset using the optimized camera trajectory during 3D reconstruction as the camera extrinsic setting. Since the camera movement during 3D acquisition is small and stable between frames, we also use the sampled key-frames from the camera trajectory during rendering. We name the sub-sample trajectory based on their frame interval n as *keyframe* n: $keyframe1$ represents that we use every frame along the trajectory during

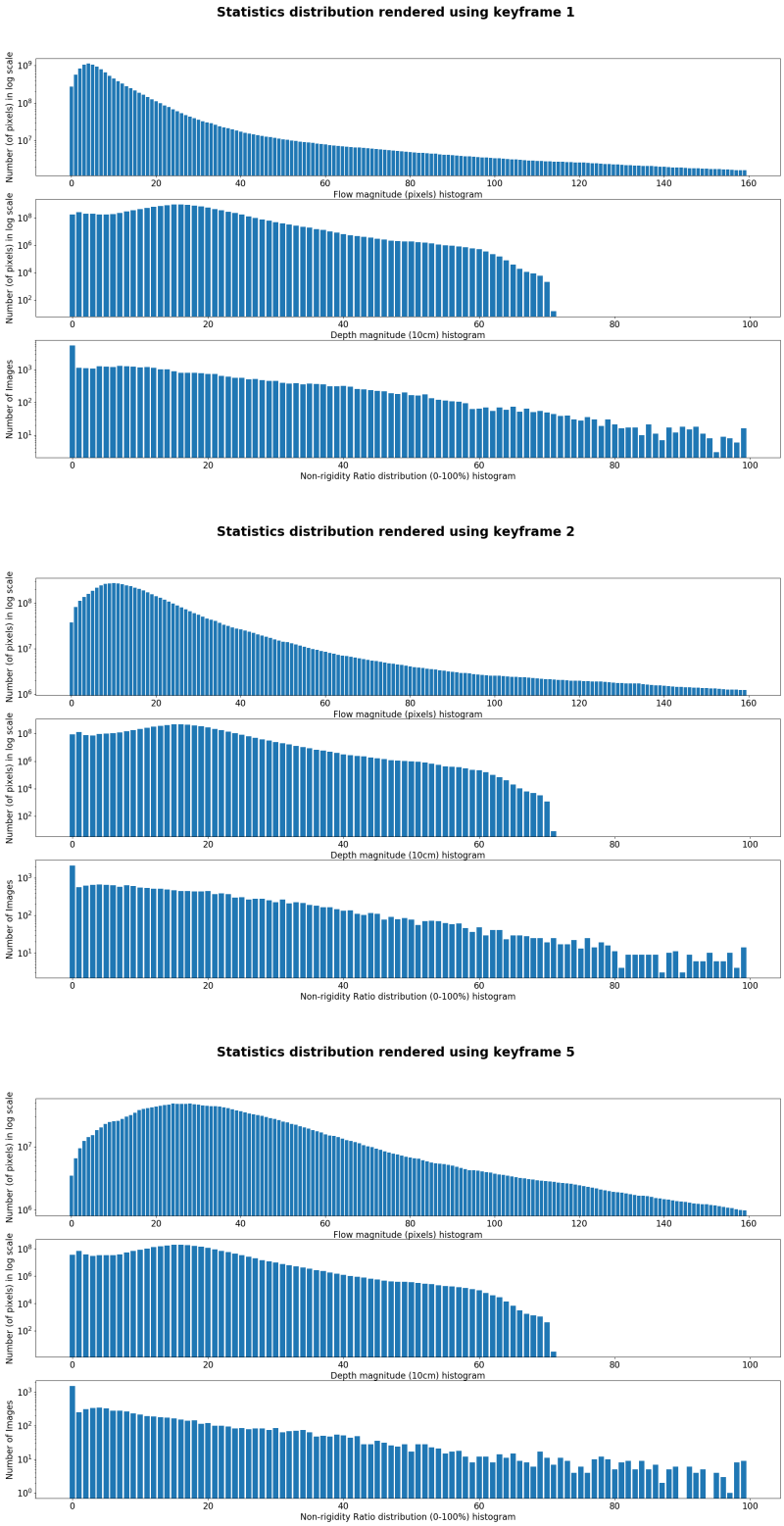


Figure 5.6: Histogram distributions of optical flow, depth, and rigidity from our rendered REFRESH dataset in the training set. We calculate the distribution from three splits using keyframes 1, 2, 5 independently. In each of the split, we show the flow magnitude distribution (top) in pixels, depth distribution (medium) in centimeters, and nonrigid ratio (belows) in the number of different images.

dataset creation and $keyframe10$ represents we use every ten frames. We list the number of static scene frames with varying keyframes in Table 5.1.

Fig. 5.6 shows the histogram distributions of our outputs in optical flow, depth, and rigidity from the rendered REFRESH dataset. We show the histogram distribution independently for the data rendered from different keyframes (1,2,5). Compare different keyframe splits, the distribution in depth and non-rigid area ratio in the images are similar and when using larger keyframes, the output optical flow tends to have a larger displacement. When using rendered outputs from larger keyframes, we can simulate the observations from a camera with larger motions.

During training, we empirically find the network generalize the best when using keyframe [1,2,5] from the optimized trajectory from BundleFusion. We use the first seven scenes in BundleFusion as our training set ('apt0', 'apt1', 'apt2', 'copyroom', 'office0', 'office1', 'office2') as our training set with a total of 69218 pairs of frames, and use 'office3' as the validation set with 6390 pairs of frames.

5.5 Experiments

We implemented the RTN in PyTorch, and the pose refinement in C++ with GTSAM 4.0. The PWCNet [48] is trained in Caffe. We integrate all the modules through Python. We use 68K images from our REFRESH dataset for training. We train RTN from scratch using weight initialization from He et al.[103] and Adam optimizer ($\beta_1 = 0.9$ and $\beta_2 = 0.999$, learning rate of $2e^{-4}$) on 3 GPUs for 12 epochs. During training, the rigidity mask loss is accumulated over 5 different scales with balanced weights, and we choose $w_\Theta = 100$. We follow the same training as PWC-net Sun et al. [48].

We evaluate our approach under various settings to show the performance of rigidity and pose estimation and their influence on scene flow estimation. For the effective analysis in scenes with different levels of non-rigid motions, we create a new test split from SINTEL data [96] based on the non-rigid number of pixels percentage. In Sec. 5.5.2, we provide a

comparison of the performance with different settings for RTN, refinement and other state-of-the-arts methods. In Sec. 5.5.3, we qualitative evaluate of our method using real world images. Please also refer to the video for more qualitative evaluations.

Datasets: We perform evaluation primarily on the challenging SINTEL dataset [96], which is a 3D rendered animation containing a sequence of 23 dynamic scenes with cinematic camera motion. For the test split, to effectively evaluate and analyze the impact of different levels of non-rigid motions in the estimation, we choose *alley_2*(1.8%), *temp_2*(5.8%), *market_5*(27.04%), *ambush_6*(38.96%), *cave_4*(47.10%), where $(\cdot)$ indicates the average non-rigid regions in each scene sequence. These examples also contain a sufficient amount of camera motion. We use the first 5 frames in the rest of the 18 scenes as a validation set, and the remaining images for training in our finetuning setting.

5.5.1 Baseline and Ablations

We evaluate our method compared to the following baselines and ablation settings.

Classical RGB-D scene flow: We compare our method to two state-of-art RGB-D scene flow solutions: SRSF [38] and VO-SF [39]. Both of these methods are optimization-based approaches that explicitly estimate the camera transform as part of the solution to flow correspondence.

Refinement only: . We denote it as solving the refinement stage without any information acquired from RTN. This strategy is often used to solve rigid motions by assuming non-rigid motions can be filtered in optimization as high-frequency outliers [6, 39].

RANSAC flow: . We use three-point RANSAC algorithm to calculate the camera pose from the flow and depth.

Semantic rigidity: To demonstrate the performance of our rigidity estimation (RTN), we compare our method to semantic rigidity estimation [89], which assumes that the non-rigid motion can be predicted from its semantic labelling. To fairly evaluate the gener-

Table 5.2: **Quantitative evaluation in flow residuals using SINTEL dataset on our test split.** The ratio of Nonrigid (NR) Region indicates the average ratio of pixels in the scene which represents the complexity of dynamic motion in the scene. We report the EPE in egomotion flow (EF) and projected scene flow (PSF). For all the baseline methods in both non-finetuning (NO FT) and finetuning (FT) setting, we use the same optical flow network trained as our method. The lowest residual under the same setting (e.g. NO FT, clean set) is highlighted as **bold**. For (f) & (p), the RTN is trained using FlyingThings dataset [31].

		NR<10%				NR 10%-40%				NR>40%		All Test	
		alley_2		temple_2		market_5		ambush_6		cave4		Average	
		EF	PSF	EF	PSF	EF	PSF	EF	PSF	EF	PSF	EF	PSF
CLEAN (no motion blur)													
	(a) SRSF [38]	4.24	7.25	7.59	16.55	25.26	31.67	17.84	37.21	10.77	11.82	12.47	18.57
	(b) VOSF [39]	6.53	1.13	5.13	10.36	16.02	35.24	13.39	28.31	6.05	9.30	8.86	15.24
NO FT	(c) Refine only	0.29	0.48	0.90	2.95	8.81	22.34	3.59	14.39	2.18	5.88	3.09	8.47
	(d) Semantic[89]+Refine	0.25	0.53	1.07	3.87	5.77	15.74	1.70	9.58	0.85	4.34	1.96	6.42
	(e) RANSAC+Flow	0.31	0.57	0.47	2.73	7.36	19.19	3.86	14.89	2.17	5.94	2.69	7.78
	(f) RTN(Things[31])+Refine	0.34	0.60	1.47	3.98	7.21	18.73	21.84	23.97	1.17	4.90	4.20	5.85
	(g) RTN(no-pose)+Refine	**0.13**	**0.45**	0.49	2.79	5.78	16.24	3.72	16.92	1.67	5.37	2.07	7.09
	(h) **RTN+Refine**	0.18	0.48	**0.46**	**2.72**	**1.61**	**11.86**	**0.97**	**8.61**	**0.63**	**4.05**	**0.74**	**5.10**
FT	(i) Semantic[89]+Refine	0.19	**0.46**	0.50	2.73	2.73	13.45	1.13	9.94	2.07	5.87	1.35	5.98
	(j) **RTN+Refine**	**0.18**	0.47	**0.42**	**2.64**	**1.69**	**11.53**	**0.47**	**7.74**	**0.91**	**4.34**	**0.77**	**5.03**
FINAL (with motion blur)													
	(k) SRSF [38]	4.33	7.78	7.59	15.51	24.93	31.29	17.26	39.08	10.80	13.29	12.37	18.86
	(l) VOSF [39]	6.29	1.54	5.69	8.91	15.99	35.17	13.37	24.02	6.23	9.28	8.96	14.61
NO FT	(m) Refine only	0.28	0.57	0.90	3.77	8.80	20.64	3.59	20.41	2.18	6.52	3.09	8.95
	(n) Semantic[89]+Refine	0.25	0.52	0.96	3.83	>100	>100	20.23	35.46	11.05	12.81	>100	>100
	(o) RANSAC+Flow	0.36	0.61	0.62	3.41	4.68	18.69	5.79	20.86	2.28	6.55	2.31	8.47
	(p) RTN(Things[31])+Refine	0.25	0.52	5.06	9.82	4.88	16.99	33.44	52.21	1.05	5.07	5.44	11.88
	(q) RTN(no-pose)+Refine	0.19	0.48	**0.82**	**3.58**	2.15	13.97	3.34	20.02	1.52	5.72	1.36	7.14
	(r) **RTN+Refine**	**0.18**	**0.47**	0.88	3.93	**0.79**	**11.87**	**2.82**	**19.42**	**0.66**	**4.66**	**0.82**	**6.29**
FT	(s) Semantic[89]+Refine	**0.19**	**0.48**	1.91	5.19	1.58	13.02	2.58	19.11	2.13	6.50	1.55	7.39
	(t) **RTN+Refine**	0.21	**0.48**	**0.66**	**3.27**	**0.97**	**11.35**	**2.34**	**19.08**	**0.74**	**4.75**	**0.79**	**6.12**

Figure 5.7: **Qualitative visualization of our results on SINTEL test split**. We compare our rigidity prediction with the output using semantic rigidity [89] trained on our REFRESH dataset and our projected scene flow with output of VOSF [39].

alization ability of semantic rigidity w.r.t our RTN, we follow Wulff et al [89] and use the DeepLab [106] architecture with weights initialized from the pre-trained MS-COCO model, but trained over the same data we used for our model. In the pose refinement stage, we substitute our rigidity from RTN with the semantic rigidity. For the fine-tuned evaluation on SINTEL, we re-train both our RTN and the semantic rigidity network. Both baselines use the same optical flow network with the same weights, and all methods use the same depth from SINTEL ground truth. We use the EPE in egomotion flow and projected scene flow defined in Section 5.2 as a metric.

Table 5.3: **Quantitative evaluation in relative camera transfrom using our SINTEL test split**. We report the relative pose error [104] (RPE) composed of translation (t) error and rotation error (r) in Euler angles (degree) in SINTEL depth metric averaged on from outputs using *clean* and *final* pass.

	NR Region <10%				NR Region 10% - 40%				NR Region >40%		All Test	
	alley_2		temple_2		market_5		ambush_6		cave4		Average	
	RPE(t)	RPE(r)	RPE(t)	RPE(r)	RPE(t)	RPE(r)	RPE(t)	RPE(r)	RPE(t)	RPE(r)	RPE(t)	RPE(r)
ORB-SLAM [105]	0.0300	0.0190	0.1740	0.0220	0.1500	0.0160	0.0550	0.0280	0.0167	0.0277	0.0894	0.0218
SRSF [38]	0.0487	0.0141	0.1763	0.0117	0.1566	0.0105	0.0672	0.0729	0.0218	0.0150	0.0980	0.0180
VOSF[39]	0.1043	0.0316	0.1055	0.0155	0.0605	**0.0006**	0.0375	0.0190	0.0438	0.0046	0.0750	0.0136
Registration [1]	0.0400	0.0094	0.3990	0.0381	0.0269	0.0073	0.0698	0.0225	0.0551	0.0076	0.1251	0.0162
RANSAC+Flow	0.0026	0.0047	0.0258	0.0033	0.0446	0.0043	0.0318	0.0082	0.0318	0.0411	0.0267	0.0039
Our RTN Pose	0.0349	0.0237	0.1589	0.0120	0.1520	0.0208	0.0455	0.0493	0.0233	0.0212	0.0883	0.0220
Ours (no ft)	**0.0015**	**0.0036**	**0.0215**	**0.0010**	**0.0059**	**0.0009**	**0.0153**	**0.0061**	**0.0053**	**0.0009**	**0.0091**	**0.0020**

Table 5.4: Evaluation of rigidity using mean IOU of rigid and nonrigid scenes.

mean IOU	REFRESH val	SINTEL clean val	SINTEL final val
Semantic Rigidity [2] trained on REFRESH	0.934	0.392	0.446
RTN trained on Things [4]	-	0.283	0.286
RTN trained on our REFRESH	**0.956**	**0.542**	**0.627**

5.5.2 Quantitative Results

Scene flow evaluation metrics: We list the end-point-error (EPE) of the ego-motion flow (EF) and projected scene flow (PSF) as defined in Section 5.2. Our proposed metric overcomes the traditional difficulty of 3D motion flow evaluation.

Scene flow evaluation: We show our quantitative evaluations using flow metric in Table 5.2, relative pose metric in Table 5.3, and the rigidity IOU in Table 5.4. The qualitative result is shown in Fig. 5.7. We can draw the following conclusions from the comparison from the results.

- Compared to classical optimization based scene flow method SRSF[38] or VOSF[39], our proposed algorithm with learned rigidity can improve scene flow accuracy by a significant margin (rows (a),(b) vs (f); (i),(j) vs (n));

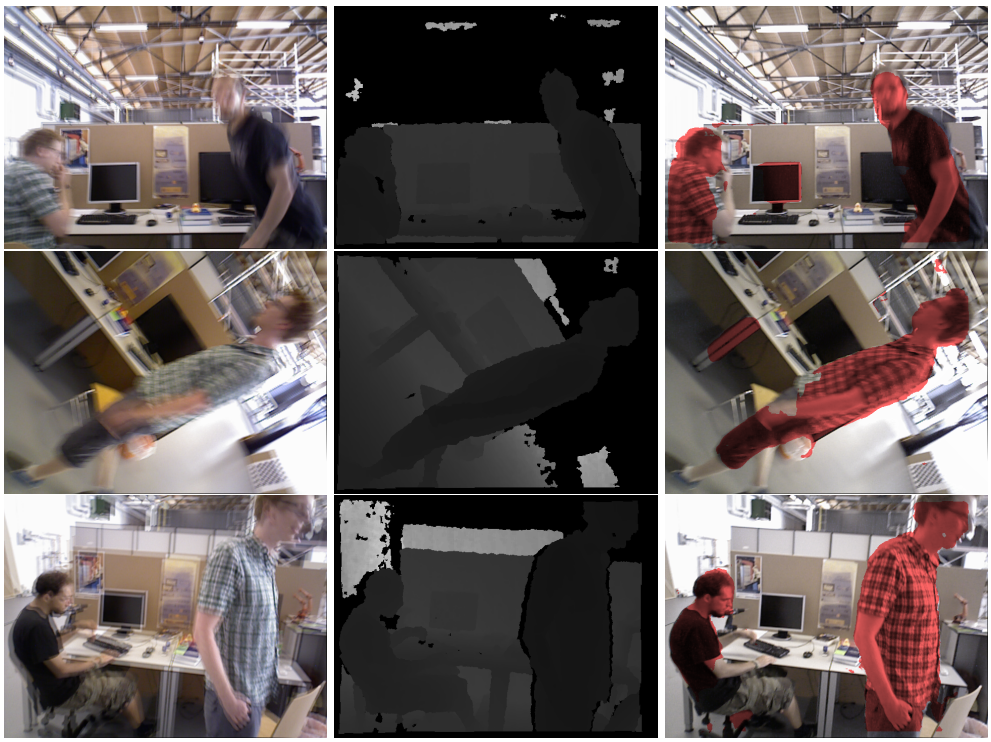
- The rigidity mask from our RTN performs better than the single-view semantic segmentation based approach [89], particularly in the more realistic *final pass* setting with no fine-tuning (NO FT) (rows (d) vs (e),(f); (l) vs (m),(n));

- As shown in *RTN+refine* setting, jointly learning rigidity with pose transform can achieve better performance than learning RTN without pose-regression (rows (e) vs (f); (m) vs (n))

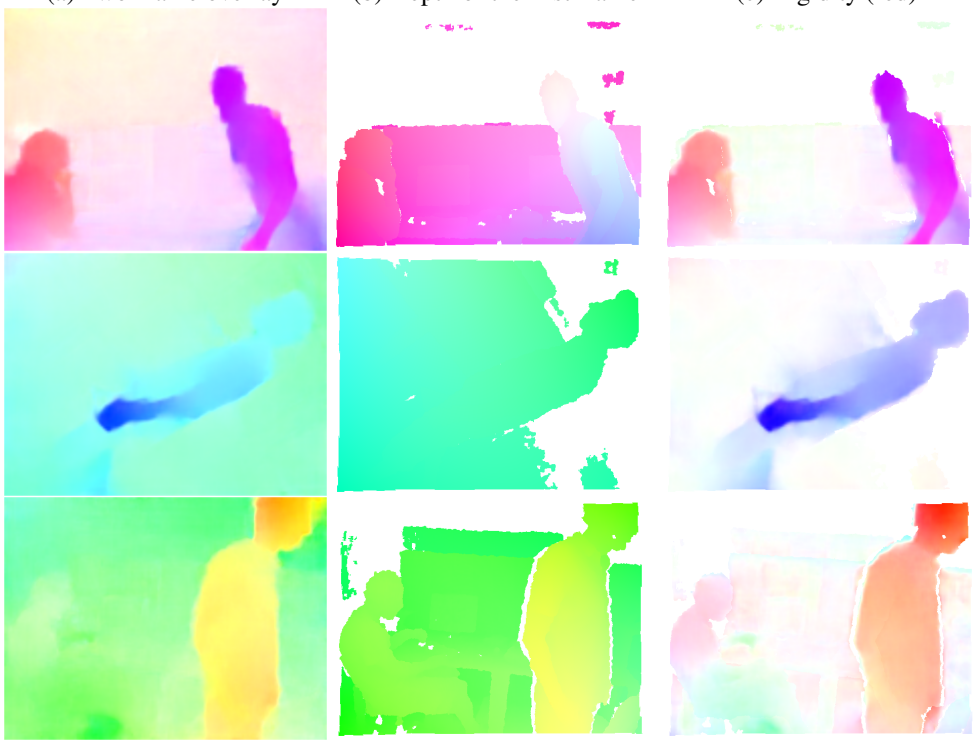
- Compared to the semantic rigidity mask [89], which relies on fine-tuning on SINTEL to achieve better performance, our learned rigidity can generalize to unseen complex scenes and perform as well as the fine-tuned model. Our rigidity prediction can capture unseen objects well, as shown by the dragon in Figure 5.7.

The Camera Pose evaluation: We include two additional baselines for pose evaluation: depth-based ORB-SLAM[105] and point cloud registration [107]. As mentioned, the accuracy of all relevant methods in dynamic scenes with moving camera highly relies on the ability ignore the non-rigid surfaces. As shown in the Table 5.3, our pose directly predicted from RTN can achieve same or better accuracy with all relevant methods, and our final solution *without fine-tunning* can out-perform all the state-of-the-art methods by a significant margin.

The Rigidity Evaluation: Table 5.4 further shows the generalization comparison in rigidity estimation. Our approach trained on our dataset generalizes significant better compared to the same approach trained using Things3D[31] and the semantic rigidity[89] using the same REFRESH data as we do.



(a) Two-frame overlay (b) Depth of the first frame (c) Rigidity (red)



(d) Optical flow (e) Ego-motion flow (f) Projected scene flow

Figure 5.8: **Qualitative visualization** of dynamic sequences in TUM [104] sequences.

Figure 5.9: **Rigidity on KITTI with network trained on our REFRESH dataset.** There is no finetuning of the network using any urban scene data.

5.5.3 Evaluation on Real-world Images

Generalization to real dynamic scene RGBD data

We use three sequences from the TUM RGB-D datasets [104] which contains dynamic motions observed from a moving Kinect camera. The depth input is noisy with missing observations and the color images contain severe motion blur. We use the raw color and depth input with provided calibrated camera intrinsics as input, and mark the regions as invalid region when the depth value is not within $[0.1, 8]$. In invalid regions, we ignore the rigidity prediction and treat the flow correspondence as outliers.

Considering there is no 3D motion flow ground truth for our real data, we visualize

the rigidity prediction and projected scene flow to qualitatively show our algorithm performance in Fig. 5.8. Our results show that our trained model on semi-synthetic data can also generalize well to real noisy RGB-D data with significant motion blur.

Generalization of rigidity mask to outdoor domain

As an interest to see how our method and the data perform in a completely different domains with above domain discrepancies, we performed a qualitative evaluation on KITTI using the same RTN network trained on our dataset and dense depth calculated from PSMnet [108] output. It is worth to note that a fair quantitative evaluation on the KITTI dataset is challenging because: (1) the available ground truth depth from LIDAR is sparse for our method, and (2) the portion of moving regions is smaller.

Fig. 5.9 shows that our RTN can generalize to KITTI reasonably well despite the domain gap and imperfect depth. We find the errors are more likely to happen in regions where the input depth uncertainty is higher and the surfaces are rigid planar, or textureless, which are not covered in our current generated data. This observation may inspire us to generate a mixture of nonrigid and rigid moving objects to improve the dataset diversity.

5.6 Discussion and Conclusion

We have presented a learning-based approach to estimate the rigid regions in dynamic scenes observed by a moving camera. Furthermore, we have shown that our framework can accurately compute the 3D motion field (scene flow), and the relative camera transform between two views. To provide better supervision to the rigidity learning task and encourage the generalization of our model, we created a novel semi-synthetic dynamic scene dataset, REFRESH, which contains real-world background scenes together with synthetic foreground moving objects. Through various tests, we have shown that our proposed method can outperform state-of-the-art solutions. We also included a new guideline for dynamic scene evaluation regarding the amount of scene motion and camera motion.

We observed some cases where the rigidity mask deviates from the ground-truth. We noticed that in these situations the moving object size is small, or the temporal motions between the two frames are small. In these cases, the error and deviations scales are small, which does not significantly affect the 3D scene flow computed as a result. Note that the success of this method also depends on the accuracy of optical flow. In scenarios when the optical flow fails or produces a noisy result, the errors in the correspondences will also propagate to 3D motion field. In future work, we can address these problems by exploiting rendering more diverse datasets to encourage generalization in different scenes. We will also incorporate both rigidity and optical flow to refine the correspondence estimation and explore performance improvements with end-to-end learning, including correspondence refinement and depth estimation from RGB inputs.

CHAPTER 6

TAKING A DEEPER LOOK AT THE INVERSE COMPOSITIONAL ALGORITHM

Summary

In this work, we provide a modern synthesis of the classic inverse compositional algorithm for dense image alignment. We first discuss the assumptions made by this wellestablished technique, and subsequently propose to relax these assumptions by incorporating data-driven priors into this model. More specifically, we unroll a robust version of the inverse compositional algorithm and replace multiple components of this algorithm using more expressive models whose parameters we train in an end-to-end fashion from data. Our experiments on several challenging 3D rigid motion estimation tasks demonstrate the advantages of combining optimization with learning-based techniques, outperforming the classic inverse compositional algorithm as well as data-driven image-to-pose regression approaches.

6.1 Introduction

Since the seminal work by Lucas and Kanade [109], dense image alignment has become an ubiquitous tool in computer vision with many applications including stereo reconstruction [110], tracking [111, 112, 113], image registration [114, 115, 116], super-resolution [117] and SLAM [118, 119, 120]. In this chapter, we provide a learning-based perspective on the Inverse Compositional algorithm, an efficient variant of the original Lucas-Kanade image registration technique. In particular, we lift some of the restrictive assumptions by parameterizing several components of the algorithm using neural networks and training the entire optimization process end-to-end.

We will first briefly review the Lucas-Kanade algorithm, the Inverse Compositional algorithm, as well as the robust M-Estimator which form the basis for our model. More details can be found in the comprehensive reviews of Baker et al. [121, 122].

The Lucas-Kanade Algorithm: The Lucas-Kanade algorithm minimizes the photometric error between a template and an image. Letting $\mathbf{T} : \Xi \rightarrow \mathbb{R}^{W \times H}$ and $\mathbf{I} : \Xi \rightarrow \mathbb{R}^{W \times H}$ denote the warped template and image, respectively[1], the Lucas-Kanade objective can be stated as follows

$$\min_{\boldsymbol{\xi}} \|\mathbf{I}(\boldsymbol{\xi}) - \mathbf{T}(\mathbf{0})\|_2^2 \tag{6.1}$$

where $\mathbf{I}(\boldsymbol{\xi})$ denotes image $\mathbf{I}$ transformed using warp parameters $\boldsymbol{\xi}$ and $\mathbf{T}(\mathbf{0}) = \mathbf{T}$ denotes the original template.

Minimizing (6.1) is a non-linear optimization task as the image $\mathbf{I}$ depends non-linearly on the warp parameters $\boldsymbol{\xi}$. The Lucas-Kanade algorithm therefore iteratively solves for the warp parameters $\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k \circ \Delta\boldsymbol{\xi}$. At every iteration k, the warp increment $\Delta\boldsymbol{\xi}$ is obtained

[1]The warping function $\mathcal{W}_{\boldsymbol{\xi}} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ might represent translation, affine 2D motion or (if depth is available) rigid or non-rigid 3D motion. To avoid clutter in the notation, we do not make $\mathcal{W}_{\boldsymbol{\xi}}$ explicit in our equations.

by linearizing

$$\min_{\Delta\boldsymbol{\xi}} \|\mathbf{I}(\boldsymbol{\xi}_k \circ \Delta\boldsymbol{\xi}) - \mathbf{T}(\mathbf{0})\|_2^2 \tag{6.2}$$

using first-order Taylor expansion

$$\min_{\Delta\boldsymbol{\xi}} \left\| \mathbf{I}(\boldsymbol{\xi}_k) + \frac{\partial \mathbf{I}(\boldsymbol{\xi}_k)}{\partial \boldsymbol{\xi}} \Delta\boldsymbol{\xi} - \mathbf{T}(\mathbf{0}) \right\|_2^2 \tag{6.3}$$

Note that the "steepest descent image" $\mathbf{J} = \partial \mathbf{I}(\boldsymbol{\xi}_k)/\partial \boldsymbol{\xi}$ needs to be recomputed at every iteration as it depends on $\boldsymbol{\xi}_k$.

To handle outliers or ambiguities (e.g., multiple motions), robust estimation [123, 78] can be used. The robust version of the Lucas-Kanade algorithm has the following objective

$$\min_{\Delta\boldsymbol{\xi}} \ \mathbf{r}_k(\Delta\boldsymbol{\xi})^T \, \mathbf{W} \, \mathbf{r}_k(\Delta\boldsymbol{\xi}) \tag{6.4}$$

where $\mathbf{r}_k(\Delta\boldsymbol{\xi}) = \mathbf{I}(\boldsymbol{\xi}_k \circ \Delta\boldsymbol{\xi}) - \mathbf{T}(\mathbf{0})$ is the residual between image $\mathbf{I}$ and template $\mathbf{T}$ at the k'th iteration, and $\mathbf{W}$ is a diagonal weight matrix that depends on the residual[2] and is chosen based on the desired robust loss function ρ [78]. The minimizer of (6.4) after linearization is obtained as the Gauss-Newton update step [124]:

$$(\mathbf{J}^T\mathbf{W}\mathbf{J})\Delta\boldsymbol{\xi} = \mathbf{J}^T\mathbf{W} \, \mathbf{r}_k(\mathbf{0}) \tag{6.5}$$

As the approximate Hessian $\mathbf{J}^T\mathbf{W}\mathbf{J}$ easily becomes ill-conditioned, a damping term is added in practice. This results in the Levenberg–Marquardt (trust-region) update equation [125]:

$$\Delta\boldsymbol{\xi} = \left(\mathbf{J}^T\mathbf{W}\mathbf{J} + \lambda \, \mathrm{diag}(\mathbf{J}^T\mathbf{W}\mathbf{J})\right)^{-1} \mathbf{J}^T\mathbf{W} \, \mathbf{r}_k(\mathbf{0}) \tag{6.6}$$

For different values of λ, the parameter update $\Delta\boldsymbol{\xi}$ varies between the Gauss-Newton direction and gradient descent. In practice, λ is chosen based on simple heuristics.

[2] We omit this dependency to avoid clutter in the notation.

Algorithm 2: The Lucas-Kanade Algorithm

while $||\mathbf{r}_k|| > \epsilon$ **do**

  $\mathbf{r}_k(\mathbf{0}) = \mathbf{T}(\mathbf{0}) - \mathbf{I}(\boldsymbol{\xi}_k)$;

  $\mathbf{J} = \partial \mathbf{I}(\boldsymbol{\xi}_k)/\partial \boldsymbol{\xi}$;

  $\Delta\boldsymbol{\xi} = (\mathbf{J}^T\mathbf{W}\mathbf{J} + \lambda\,\mathrm{diag}(\mathbf{J}^T\mathbf{W}\mathbf{J}))^{-1}\mathbf{J}^T\mathbf{W}\,\mathbf{r}_k(\mathbf{0})$;

  $\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k \circ \Delta\boldsymbol{\xi}$

end

Inverse Compositional Algorithm: The *inverse compositional* (IC) algorithm [121] avoids the repeated calculation of the $\partial \mathbf{I}(\boldsymbol{\xi}_k)/\partial \boldsymbol{\xi}$ by applying the warp increments $\Delta\boldsymbol{\xi}$ to the template instead of the image

$$\min_{\Delta\boldsymbol{\xi}} \|\mathbf{I}(\boldsymbol{\xi}_k) - \mathbf{T}(\Delta\boldsymbol{\xi})\|_2^2 \tag{6.7}$$

using the warp parameter update $\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k \circ (\Delta\boldsymbol{\xi})^{-1}$. In the linearized equation

$$\min_{\Delta\boldsymbol{\xi}} \left\| \mathbf{I}(\boldsymbol{\xi}_k) - \mathbf{T}(\mathbf{0}) - \frac{\partial \mathbf{T}(\mathbf{0})}{\partial \boldsymbol{\xi}} \Delta\boldsymbol{\xi} \right\|_2^2 \tag{6.8}$$

where $\mathbf{J} = \partial \mathbf{T}(\mathbf{0})/\partial \boldsymbol{\xi}$ is the Jacobian of the template $\mathbf{T}(\mathbf{0})$ with respect to the warp parameters $\boldsymbol{\xi}$, which does not depend on $\boldsymbol{\xi}_k$ and can thus be pre-computed.

Algorithm 3: The Inverse Compositional Lucas-Kanade Algorithm

$\mathbf{J} = \partial \mathbf{T}(\mathbf{0})/\partial \boldsymbol{\xi}$; `// Pre-compute Jacobian`

while $||\mathbf{r}_k|| > \epsilon$ **do**

  $\mathbf{r}_k(\mathbf{0}) = \mathbf{I}(\boldsymbol{\xi}_k) - \mathbf{T}(\mathbf{0})$;

  $\Delta\boldsymbol{\xi} = (\mathbf{J}^T\mathbf{W}\mathbf{J} + \lambda\,\mathrm{diag}(\mathbf{J}^T\mathbf{W}\mathbf{J}))^{-1}\mathbf{J}^T\mathbf{W}\,\mathbf{r}_k(\mathbf{0})$;

  $\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k \circ (\Delta\boldsymbol{\xi})^{-1}$

end

The Limitations of Inverse Compositional Algorithm: Despite its widespread utility, the IC method suffers from a number of important limitations. First, it assumes that the linearized residual leads to an update which iteratively reaches a good local optimum. However, this assumption is invalid in the presence of high-frequency textural information or noise in $\mathbf{I}$ or $\mathbf{T}$. Second, choosing a good function ρ is difficult as the true data distribution is often unknown. Moreover, Equation (6.4) does not capture correlations or higher-order statistics in the inputs $\mathbf{I}$ and $\mathbf{T}$ as the residuals operate directly on the pixel values and the weight matrix $\mathbf{W}$ is diagonal. Finally, damping heuristics do not fully exploit the information available during optimization and thus lead to suboptimal solutions.

Contributions: We propose to combine the best of both (optimization and learning-based) worlds by unrolling the robust IC algorithm into a more general parameterized feed-forward model which is trained end-to-end from data. In contrast to generic neural network estimators, this allows our algorithm to incorporate knowledge about the structure of the problem (e.g., family of warping functions, 3D geometry) as well as the advantages of a robust iterative estimation framework. At the same time, our approach relaxes the restrictive assumptions made in the original IC formulation [121] by incorporating trainable modules and learning the entire model end-to-end.

More specifically, we make the following contributions:

(A) We propose a **Two-View Feature Encoder** which replaces $\mathbf{I}, \mathbf{T}$ with feature representations $\mathbf{I}_\theta, \mathbf{T}_\theta$ that jointly encode information about the input image $\mathbf{I}$ and the template $\mathbf{T}$. This allows our model to exploit spatial as well as temporal correlations in the data.

(B) We propose a **Convolutional M-Estimator** that replaces $\mathbf{W}$ in (6.4) with a learned weight matrix $\mathbf{W}_\theta$ which encodes information about $\mathbf{I}$, $\mathbf{T}$ and $\mathbf{r}_k$ in a way such that the unrolled optimization algorithm ignores irrelevant or ambiguous information as well as outliers.

(C) We propose a **Trust Region Network** which replaces the damping matrix $\lambda \operatorname{diag}(\mathbf{J}^T\mathbf{W}\mathbf{J})$ in (6.6) with a learned damping matrix $\operatorname{diag}(\boldsymbol{\lambda}_\theta)$ whose diagonal entries λ_θ are estimated from "residual volumes" which comprise residuals of a trust-region update when applying a range of hypothetical λ values.

We demonstrate the advantages of combining the classical IC method with deep learning on the task of 3D rigid motion estimation using several challenging RGB-D datasets.

6.2 Related Work

We are not the first to inject deep learning into an optimization pipeline. In this section, we first review classical methods, followed by direct pose regression techniques and related work on learning-based optimization.

Classical Methods: Direct methods [126, 127] that align images using the sum-of-square error objective (6.1) are prone to outliers and varying illuminations. Classical approaches address this issue by exploiting more robust objective functions [109, 128], heuristically chosen patch- [129] or gradient-based [115] features, and photometric calibration as a pre-processing step [120]. The most common approach is to use robust estimation (6.4) as in [130]. However, the selection of a good robust function is challenging and traditional formulations assume that the same function applies to all pixels, ignoring correlations in the inputs. Moreover, the inversion of the linearized system (6.5) may still be ill-conditioned [131]. To overcome this issue, soft constraints in the form of damping terms (6.6) are added to the objective [121, 122]. However, this may bias the system to sub-optimal solutions.

This paper addresses these problems by relaxing the main assumptions of the Inverse Compositional (IC) algorithm [121, 122] using data-driven learning. More specifically, we propose to learn the feature representation (A), robust estimator (B) and damping (C) jointly to replace the traditional heuristic rules of classical algorithms.

Direct Pose Regression: A notably different approach to classical optimization techniques

is to directly learn the entire mapping from the input to the warping parameters $\boldsymbol{\xi}$ from large amounts of data, spanning from early work using linear hyperplane approximation [132] to recent work using deep neural networks [54, 11, 57, 60, 75]. Prominent examples include single image to camera pose regression [49, 50], image-based 3D object pose estimation [52, 53] and relative pose prediction from two views [54, 11]. However, learning a direct mapping requires high-capacity models and large amounts of training data. Furthermore, obtaining pixel-accurate registrations remains difficult and the learned representations do not generalize well to new domains. To improve accuracy, recent methods adopt cascaded networks [133, 46] and iterative feedback [134]. Lin et al. [135] combines the multi-step iterative spatial transformer network (STN) with the classical IC algorithm [121, 122] for aligning 2D images. Variants of this approach have recently been applied to various 3D tasks: Li et al. [136] proposes to iteratively align a 3D CAD model to an image. Zhou et al. [55] jointly train for depth, pose and optical flow.

Different from [135] and its variants which approximate the pseudo-inverse of the Jacobian implicitly using stacked convolutional layers, we exploit the structure of the optimization problem and explicitly solve the original robust objective (6.4) with learned modules using few parameters.

Learning-based Optimization: Recently, several methods have exploited the differentiable nature of iterative optimization algorithms by unrolling for a fixed number of iterations. Each iteration is treated as a layer in a neural network [**Zsheng15iccv**, 137, 138, 139, 140, 141]. In this section, we focus on the most related work which also tackles the least-squares optimization problem [75, 118, 142, 143]. We remark that most of these techniques can be considered special cases of our more general deep IC framework.

Wang et al. [111] address the 2D image tracking problem by learning an input representation using a two-stream Siamese network for the IC setting. In contrast to us, they exploit only spatial but not temporal correlations in the inputs (A), leverage a formulation which is not robust (B) and do not exploit trust-region optimization (C).

Clark et al. [118] propose to jointly learn depth and pose estimation by minimizing photometric error using (6.7). In contrast to us, they do not learn feature representations (A) and neither employ a robust formulation (B) nor trust-region optimization (C).

Ranftl et al. [142] propose to learn robust weights for *sparse* feature correspondences and apply their model to fundamental matrix estimation. As they do not target direct image alignment, their problem setup is different from ours. Besides, they neither learn input features (A) nor leverage trust-region optimization (C). Instead, they solve their optimization problem using singular value decomposition.

Concurrent to our work, Tang and Tan [143] propose a photometric Bundle Adjustment network by learning to align feature spaces for monocular reconstruction of a static scene. Different from us, they did not exploit temporal correlation in the inputs (A) and do not employ a robust formulation (B). While they propose to learn the damping parameters (C), in contrast to our trust-region volume formulation, they regress the damping parameters from the global average pooled residuals.

6.3 Method

This section describes our model. A high-level overview over the proposed unrolled inverse-compositional algorithm is given in Fig. 6.1. Using the same notation as in Section 6.1, our goal is to minimize the error by warping the image towards the template, similar to (6.4):

$$\min_{\boldsymbol{\xi}} \; \mathbf{r}(\boldsymbol{\xi})^T \, \mathbf{W}_\theta \, \mathbf{r}(\boldsymbol{\xi}) \tag{6.9}$$

The difference to (6.4) is that in our formulation, the weight matrix $\mathbf{W}_\theta$ as well as the template $\mathbf{T}_\theta(\boldsymbol{\xi})$ and the image $\mathbf{I}_\theta(\boldsymbol{\xi})$ (and thus also the residual $\mathbf{r}(\boldsymbol{\xi}) = \mathbf{I}_\theta(\boldsymbol{\xi}) - \mathbf{T}_\theta(\mathbf{0})$) depend on the parameters of a learned model θ.

We exploit the inverse compositional algorithm to solve the non-linear optimization

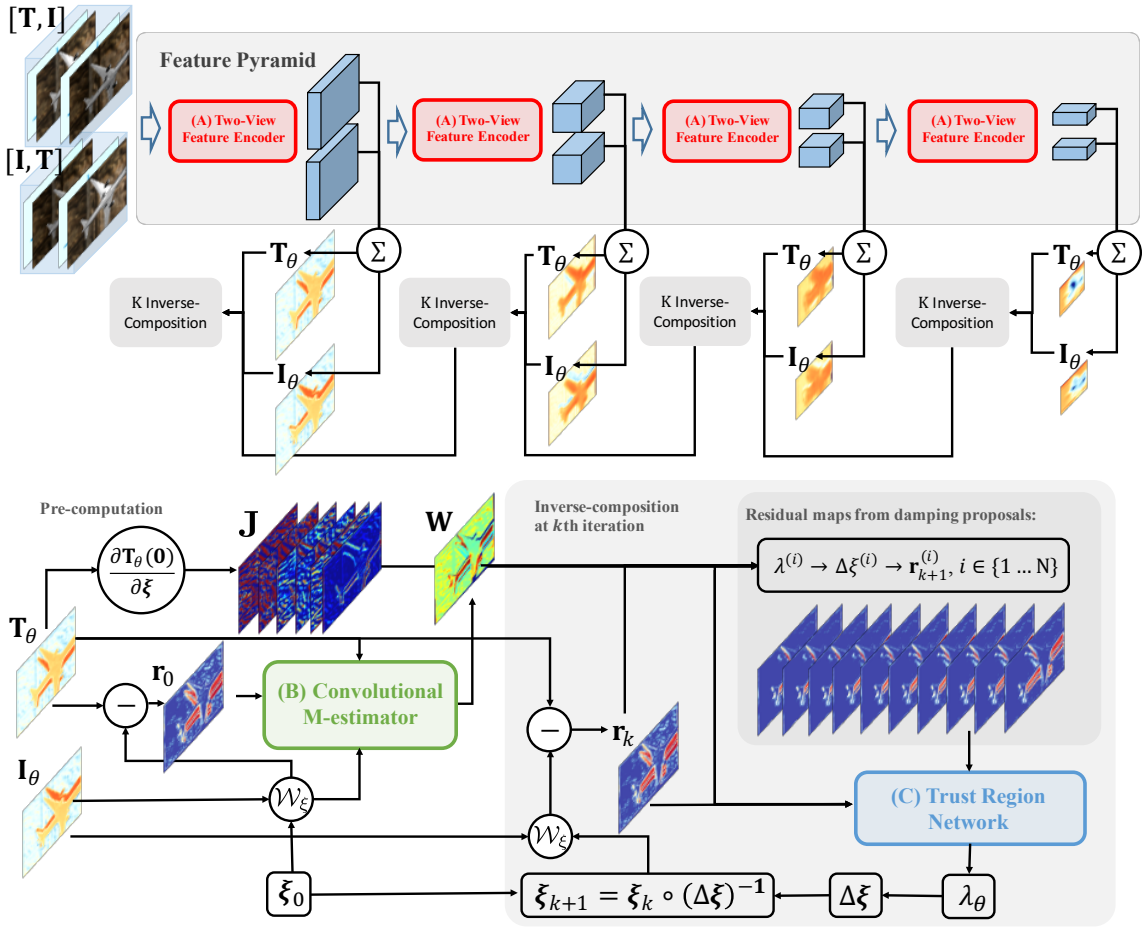


Figure 6.1: **High-level Overview of our Deep Inverse Compositional (IC) Algorithm.** We stack $[\mathbf{I}, \mathbf{T}]$ and $[\mathbf{T}, \mathbf{I}]$ as inputs to our **(A) Two-View Feature Encoder** pyramid which extracts 1-channel feature maps $\mathbf{T}_\theta$ and $\mathbf{I}_\theta$ at multiple scales using channel-wise summation. We then perform K IC steps at each scale using $\mathbf{T}_\theta$ and $\mathbf{I}_\theta$ as input. In each scale, we pre-compute $\mathbf{W}$ using our **(B) Convolutional M-estimator**. For each of the K IC iterations, we compute the warped image $\mathbf{I}(\boldsymbol{\xi}_k)$ and $\mathbf{r}_k$. Subsequently, we sample N damping proposals $\lambda^{(i)}$ and compute the proposed residual maps $\mathbf{r}_{k+1}^{(i)}$. Our **(C) Trust Region Network** takes these residual maps as input and predicts λ_θ for the trust region update step.

problem in (6.9), i.e., we linearize the objective and iteratively update the warp parameters

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k \circ (\Delta\boldsymbol{\xi})^{-1} \tag{6.10}$$

$$\Delta\boldsymbol{\xi} = (\mathbf{J}^T\mathbf{W}_\theta\mathbf{J} + \text{diag}(\boldsymbol{\lambda}_\theta))^{-1}\mathbf{J}^T\mathbf{W}_\theta\,\mathbf{r}_k(\mathbf{0}) \tag{6.11}$$

$$\mathbf{r}_k = \mathbf{I}_\theta(\boldsymbol{\xi}_k) - \mathbf{T}_\theta(\mathbf{0}) \tag{6.12}$$

$$\mathbf{J} = \partial\mathbf{T}_\theta(\mathbf{0})/\partial\boldsymbol{\xi} \tag{6.13}$$

starting from $\boldsymbol{\xi}_0 = \mathbf{0}$.

Algorithm 4: Deeper Inverse Compositional Algorithm with N iterations

$\mathbf{J} = \partial\mathbf{T}_\theta(\mathbf{0})/\partial\boldsymbol{\xi}$; `// Pre-compute Jacobian`

while $k < N$ **do**

  $\mathbf{r}_k = \mathbf{I}_\theta(\boldsymbol{\xi}_k) - \mathbf{T}_\theta(\mathbf{0})$;

  $\Delta\boldsymbol{\xi} = (\mathbf{J}^T\mathbf{W}_\theta\mathbf{J} + \text{diag}(\boldsymbol{\lambda}_\theta))^{-1}\mathbf{J}^T\mathbf{W}_\theta\,\mathbf{r}_k(\mathbf{0})$;

  $\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k \circ (\Delta\boldsymbol{\xi})^{-1}$

end

The most notable change from (6.6) is that the image features $(\mathbf{I}_\theta, \mathbf{T}_\theta)$, the weight matrix $(\mathbf{W}_\theta)$ and the damping factors $\boldsymbol{\lambda}_\theta$ are predicted by learned functions which have been collectively parameterized by $\theta = \{\theta_\mathbf{I}, \theta_\mathbf{W}, \theta_{\boldsymbol{\lambda}}\}$. Note that also the residual $\mathbf{r}_k$ as well as the Jacobian $\mathbf{J}$ implicitly depend on the parameters θ, though we have omitted this dependence here for notational clarity. We will now provide details about these mappings.

(A) Two-View Feature Encoder: We use a fully convolutional neural network ϕ_θ to extract feature maps from the image $\mathbf{I}$ and the template $\mathbf{T}$:

$$\mathbf{I}_\theta = \phi_\theta([\mathbf{I}, \mathbf{T}]) \tag{6.14}$$

$$\mathbf{T}_\theta = \phi_\theta([\mathbf{T}, \mathbf{I}]) \tag{6.15}$$

Here, the operator $[\cdot,\cdot]$ indicates concatenation along the feature dimension. Instead of $\mathbf{I}$ and $\mathbf{T}$ we then feed $\mathbf{I}_\theta$ and $\mathbf{T}_\theta$ to the residuals in (6.12) and to the Jacobian in (6.13). Note that we use the notation $\mathbf{I}_\theta(\boldsymbol{\xi}_k)$ in (6.12) to denote that the feature map $\mathbf{I}_\theta$ is warped by a warping function that is parameterized via $\boldsymbol{\xi}$. More formally, this can be stated as $\mathbf{I}_\theta(\boldsymbol{\xi}) = \mathbf{I}_\theta(\mathcal{W}_{\boldsymbol{\xi}}(\mathbf{x}))$, where $\mathbf{x} \in \mathbb{R}^2$ denote pixel locations and $\mathcal{W}_{\boldsymbol{\xi}} : \mathbb{R}^2 \to \mathbb{R}^2$ is a warping function that maps a pixel location to another pixel location. For instance, $\mathcal{W}_{\boldsymbol{\xi}}$ may represent the space of 2D translations or affine transformations. In our experiments, we will focus on challenging 3D rigid body motions, using RGB-D inputs and representing $\boldsymbol{\xi}$ as an element of the Euclidean group $\boldsymbol{\xi} \in \mathfrak{se}(3)$.

Note that compared to directly using the image $\mathbf{I}$ and $\mathbf{T}$ as input, our features capture high-order *spatial* correlations in the data, depending on the receptive field size of the convolutional network. Moreover, they also capture *temporal* information as they operate on both $\mathbf{I}$ and $\mathbf{T}$ as input.

(B) Convolutional M-Estimator: We parameterize the weight matrix $\mathbf{W}_\theta$ as a diagonal matrix whose elements are determined by a fully convolutional network ψ_θ that operates on the feature maps and the residual:

$$\mathbf{W}_\theta = \text{diag}(\psi_\theta(\mathbf{I}_\theta(\boldsymbol{\xi}_k), \mathbf{T}_\theta(\mathbf{0}), \mathbf{r}_k)) \tag{6.16}$$

as is illustrated in Fig. 6.1. Note that this enables our algorithm to reason about relevant image information while capturing spatial-temporal correlations in the inputs which is not possible with classical M-Estimators. Furthermore, we do not restrict the implicit robust function ρ to a particular error model, but instead condition ρ itself on the input. This allows for learning more expressive noise models.

(C) Trust Region Network: For estimating the damping $\boldsymbol{\lambda}_\theta$ we use a fully-connected network as illustrated in Fig. 6.1. We first sample a set of scalar damping proposals λ_i on a

logarithmic scale and compute the resulting Levenberg-Marquardt update step as

$$\Delta\boldsymbol{\xi}_i = \left(\mathbf{J}^T\mathbf{W}\mathbf{J} + \lambda_i \operatorname{diag}(\mathbf{J}^T\mathbf{W}\mathbf{J})\right)^{-1}\mathbf{J}^T\mathbf{W}\,\mathbf{r}_k(\mathbf{0}) \tag{6.17}$$

We stack the resulting N residual maps

$$\mathbf{r}_{k+1}^{(i)} = \mathbf{I}_\theta(\boldsymbol{\xi}_k \circ (\Delta\boldsymbol{\xi}_i)^{-1}) - \mathbf{T}_\theta(\mathbf{0}) \tag{6.18}$$

into a single feature map, flatten it, and pass it to a fully connected neural network ν_θ that outputs the damping parameters $\boldsymbol{\lambda}_\theta$:

$$\boldsymbol{\lambda}_\theta = \nu_\theta\left(\mathbf{J}^T\mathbf{W}_\theta\mathbf{J}, \left[\mathbf{J}^T\mathbf{W}_\theta\mathbf{r}_{k+1}^{(1)}, \ldots, \mathbf{J}^T\mathbf{W}_\theta\mathbf{r}_{k+1}^{(N)}\right]\right) \tag{6.19}$$

The intuition behind our trust region networks is that the residuals predicted using the Levenberg-Marquardt update comprise valuable information about the damping parameter itself. This is empirically confirmed by our experiments.

Coarse-to-Fine Estimation: To handle large motions, it is common practice to apply direct methods in a coarse-to-fine fashion. We apply our algorithm at four pyramid levels with three iterations each. Our entire model including coarse-to-fine estimation is illustrated in Fig. 6.1. We extract features at all four scales using a single convolutional neural network with spatial average pooling between pyramid levels. We start with $\boldsymbol{\xi} = \mathbf{0}$ at the coarsest pyramid level, perform 3 iterations of our deep IC algorithm, and proceed with the next level until we reach the original image resolution.

Training and Inference: For training and inference, we unroll the iterative algorithm in equations (6.10)-(6.13). We obtain the gradients of the resulting computation graph using auto-differentiation. For estimating the parameters θ in our unrolled optimization algorithm we leverage ADAM [144].

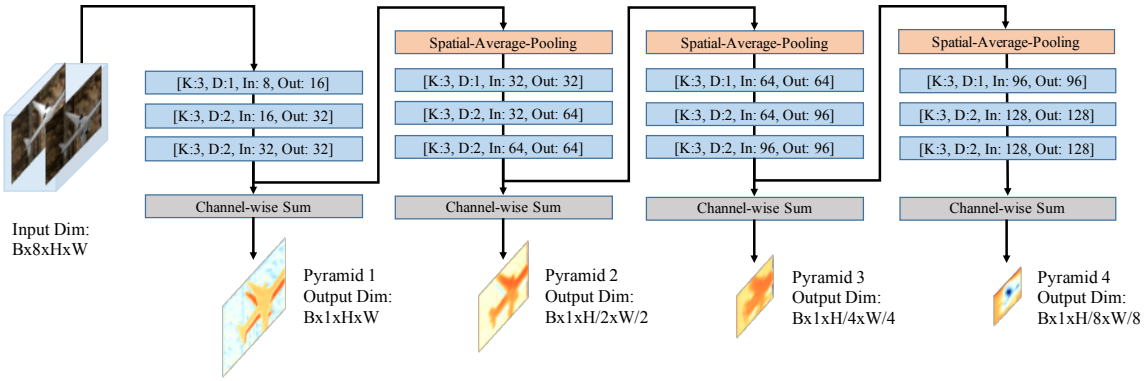


Figure 6.2: **(A) Two-view Feature Encoder**. We use [K, D, In, Out] as abbreviation for [Kernel size, Dilation, Input channel size, Output channel size]. All convolutional layers are followed by a BatchNorm layer and a ReLU layer. We use [B,H,W] as abbreviation for the feature size [Batch size, Height of feature, Width of feature]. We use spatial average pooling of size 2 to downsample features between two feature pyramids. We channel-wise sum the output features of the encoder at each scale to obtain the resulting feature maps.

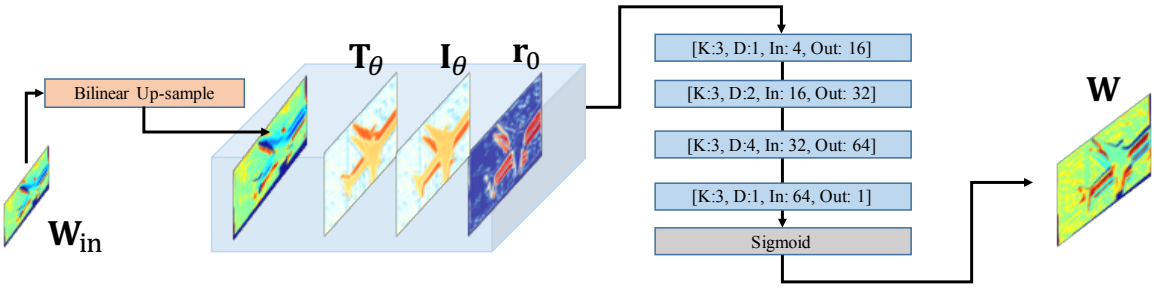


Figure 6.3: **(B) Convolutional M-Estimator**. We use [K, D, In, Out] as abbreviation for [Kernel size, Dilation, Input channel size, Output channel size]. All convolutional layers are followed by a BatchNorm layer and a ReLU layer. In our default weight-sharing setting, all weights are shared across networks in different pyramids. At the coarsest image level which does not require the up-sampled $\mathbf{W}$ as input, we set $\mathbf{W}_{\text{in}}$ to $\mathbf{1}$.

6.4 Network Details

In the following, we describe the details of the network architectures used in our model.

(A) Two-view Feature Encoder: Fig. 6.2 shows the architecture of our two-view feature encoder for estimating both $\mathbf{T}_\theta$ and $\mathbf{I}_\theta$. The network takes two concatenated RGB-D views as input. For the depth channel, we use the inverse depth d clamped to [0, 10]. For the 2D affine experiments we use only the RGB channels.

The feature pyramid comprises one (A) Two-view Feature Encoder at each of the four

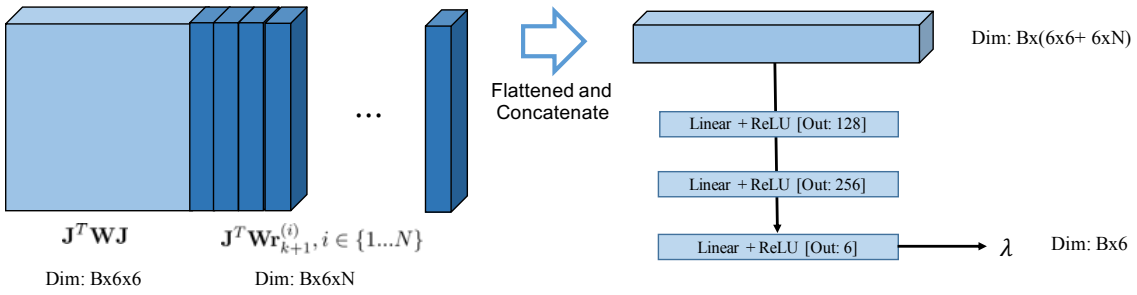


Figure 6.4: **(C) Trust Region Network.** We use B as abbreviation for Batch size. N indicates the number of damping proposals. In the weight-sharing setting, all weights are shared across networks at different pyramid levels. We use 'Linear' to represent a fully connected layer. The last ReLU layer ensures that the output λ is non-negative.

pyramid levels. Each feature encoder uses three dilated convolutional layers. We use a Spatial Average Pooling layer to downsample the output features from the fine scale as input to the next coarser scale.

(B) Convolutional M-Estimator: Fig. 6.3 shows the operations and parameters of the Convolutional M-Estimator we use in this paper. In the coarse-to-fine inverse compositional refinement, we add one more input to the network which is the predicted weight from the coarser level pyramid. At each image pyramid, we bilinear upsample the weight matrix predicted from the coarse scale $\mathbf{W_{in}}$, and concatenate it with $\mathbf{T}_\theta$, $\mathbf{I}_\theta$ and $\mathbf{r}_0$, which we use as input to the Convolutional M-estimator. The network predicts $\mathbf{W}$ at the current scale. Different from traditional M-estimators which evaluate $\mathbf{W}$ at every step when $\mathbf{r}_k$ is updated, we only compute $\mathbf{W}$ once for all following K iterations. This way, we approximate the classical M-estimator and significantly reduce computation. The network is composed of four convolutional layers, with dilation [1,2,4,1], followed by a sigmoid layer which normalizes the output to the range [0,1]. In the default setting, we use weight-sharing network as the module at different pyramid scales. Our Convolutional M-estimator is relatively small which makes inference fast. Note that despite the small size of our network, the dilation layers and the coarse-to-fine process ensure a sufficiently large receptive field.

(C) Trust Region Network: Fig. 6.4 shows the operations and parameters of our Trust

Region Network. Given the N residual maps $\mathbf{r}_{(i)}^{k}, i \in \{1...N\}$, we first calculate the right-hand-side (RHS) vector $\mathbf{J}^T\mathbf{W}\mathbf{r}_k^{(i)} \in \mathbb{R}^{1\times 6}$ corresponding to each residual map $\mathbf{r}_k^{(i)}$. Next, we flatten the N RHS vectors jointly with the approximate Hessian matrix $\mathbf{J}^T\mathbf{W}\mathbf{J} \in \mathbb{R}^{6\times 6}$ into a single vector, which is the input to our Trust Region Network. This network is composed of three fully connected layers and outputs the damping vector. At the last layer, a ReLU ensures non-negative elements. In practice, we add a small epsilon ($1e^{-5}$) to this vector, which ensure the damping matrix is a positive definite matrix.

6.5 RGB-D 3D Motion Experiments

We perform our experiments on the challenging task of 3D rigid body motion estimation from RGB-D inputs. Apart from the simple scenario of purely static scenes where only the camera is moving, we also consider scenes where both the camera as well as objects are in motion, hence resulting in strong ambiguities. We cast this as a supervised learning problem: using the ground truth motion, we train the models to resolve these ambiguities by learning to focus either on the foreground or the background.

Warping Function: Given pixel $\mathbf{x} \in \mathbb{R}^2$, camera intrinsics $\mathbf{K}$ and depth $D(\mathbf{x})$, we define the warping $\mathcal{W}_{\boldsymbol{\xi}}(\mathbf{x})$ induced by rigid body transform $\mathbf{T}_{\boldsymbol{\xi}}$ with $\boldsymbol{\xi} \in \mathfrak{se}(3)$ as

$$\mathcal{W}_{\boldsymbol{\xi}}(\mathbf{x}) = \mathbf{K}\,\mathbf{T}_{\boldsymbol{\xi}}\,D(\mathbf{x})\mathbf{K}^{-1}\,\mathbf{x} \tag{6.20}$$

Using the warped coordinates, compute $\mathbf{I}_\theta(\boldsymbol{\xi})$ via bilinear sampling from $\mathbf{I}_\theta$ and set the warped feature value to zero for all occluded areas (estimated via z-buffering).

Training Objective: To balance the influences of translation and rotation we follow [136] and exploit the 3D End-Point-Error (EPE) as loss function. Let $\mathbf{p} = D(\mathbf{x})\mathbf{K}^{-1}\mathbf{x}$ denote the 3D point corresponding to pixel $\mathbf{x}$ in image $\mathbf{I}$ and let $\mathcal{P}$ denote the set of all such 3D points.

We minimize the following loss function

$$\mathbb{L} = \frac{1}{|\mathcal{P}|} \sum_{l \in \mathcal{L}} \sum_{\mathbf{p} \in \mathcal{P}} \| \mathbf{T}_{gt}\, \mathbf{p} - \mathbf{T}(\boldsymbol{\xi}_l)\, \mathbf{p} \|_2^2 \tag{6.21}$$

where $\mathcal{L}$ denotes the set of coarse-to-fine pyramid levels (we apply our loss at the final iteration of every pyramid level) and $\mathbf{T}_{gt}$ is the ground truth transformation.

Implementation: We use the Sobel operator to compute the gradients in $\mathbf{T}$ and analytically derive $\mathbf{J}$. We calculate the matrix inverse on the CPU since we observed that inverting a small dense matrices $\mathbf{H} \in \mathbb{R}^{6\times6}$ is significantly faster on the CPU than on the GPU. In all our experiments we use $N = 10$ damping proposals for our Trust Region Network, sampled uniformly in logscale. We use four coarse-to-fine pyramid levels with three iterations each. We implemented our model and the baselines in PyTorch. All experiments start with a fixed learning rate of 0.0002 using ADAM [144]. We train a total of 15 epochs, reducing the learning rate by half every 5 epochs. From all epochs, we select the final model based on validation performance.

6.5.1 Datasets Descriptions

We systematically train and evaluate our method on the followng four datasets.

MovingObjects3D: For the purpose of systematically evaluating highly varying object motions, we downloaded six categories of 3D models from ShapeNet [145]. For each object category, we rendered 200 video sequences with 100 frames in each sequence using Blender. We use data rendered from the categories 'boat' and 'motorbike' as test set and data from categories 'aeroplane', 'bicycle', 'bus', 'car' as training set. From the training set we use the first 95% of the videos for training and the remaining 5% for validation. In total, we obtain 75K images for training, 5K images for validation, and 25K for testing. We further subsample the sequences using sampling intervals $\{1, 2, 4\}$ in order to obtain small, medium and large motion subsets.

For each rendered sequence, we randomly select one 3D object model within the chosen category and stage it in a static 3D cuboid room with random wall textures and four point light sources. We randomly choose the camera viewpoint, point light source position and object trajectory, see supplementary material for details. This ensures diversity in object motions, textures and illumination. The videos also contain frames where the object is only partially visible. We exclude all frames where the entire object is not visible.

BundleFusion: To evaluate camera motion in a static environment, we use the eight publicly released scenes from BundleFusion[3] [95] which provide fully synchronized RGB-D sequences. We hold out the entire 'copyroom' and 'office2' scenes as test set and split the remaining six scenes into training (first 95% of each trajectory) and validation (last 5% of each trajectory). We use the released camera trajectories as ground truth. We subsample frames at intervals $\{2, 4, 8\}$ to increase motion magnitudes and hence the level of difficulty.

DynamicBundleFusion (REFRESH): To further evaluate camera motion estimation under heavy occlusion and motion ambiguity, we use the DynamicBundleFusion dataset [146], which is the same dataset created by REFRESH in Section 5.4. It augments the scenes from BundleFusion with non-rigidly moving human subjects as distractors. We use the same training, validation and test split as above. We train and evaluate frames subsampled at intervals $\{1, 2, 5\}$ due to the increased difficulty of this task.

TUM RGBD SLAM: To evaluate camera motion estimation with ground truth data, we use RGBD SLAM dataset [104]. Our evaluation split of the TUM RGB-D SLAM dataset [104] consists of four trajectories of different conditions, trajectory length and motion magnitudes. After synchronization of the color image, depth and the ground truth trajectory, we obtain 750 frames in 'fr1/360', 584 frames in 'fr1/desk', 2203 frames in 'fr2/desk' and 830 frames in 'fr2/pioneer_360'. We split the remaining trajectories into training (first 95% of each trajectory) and validation (last 5% of each trajectory). We subsample the frames at intervals of $\{2,4,8\}$ to increase the motion magnitude.

[3] http://graphics.stanford.edu/projects/bundlefusion/

6.5.2 Baselines

We implemented the following baselines.

ICP: We use classical *Point-to-Plane ICP* [147] and *Point-to-Point-ICP*[148] implemented in Open3D [149]. To examine the effect of ambiguity between the foreground and background in the object motion estimation task, we also evaluate a version for which we provide the ground truth instance mask to both methods. Note that this is an upper bound to the performance achievable by ICP methods. We thus call them the *Oracle* ICP methods.

RGB-D Visual Odometry: We compare to the RGB-D visual odometry method [150] implemented in Open3D [149] on TUM RGBD datasets for visual odometry comparison.

Direct Pose Regression: We compare three different variants that directly predict the mapping $f : \mathbf{I}, \mathbf{T} \rightarrow \boldsymbol{\xi}$. All three networks use Conv1-6 encoder layers from FlowNet-Simple[19] as two-view regression backbones. We use spatial average pooling after the last feature layer followed by a fully-connected layer to regress $\boldsymbol{\xi}$. All three CNN baselines are trained using the loss in (6.21).

- **PoseCNN**: A feed-forward CNN that directly predicts $\boldsymbol{\xi}$.
- **IC-PoseCNN**: A PoseCNN with iterative refinement using the IC algorithm, similar to [135] and [136]. We noticed that training becomes unstable and performance saturates when increasing the number of iterations. For all our experiments, we thus used three iterations.
- **Cascaded-PoseCNN**: A cascaded network with three iterations, similar to IC-PoseCNN but with independent weights for each iteration.

Learning-based Optimization: We implemented the following related algorithms within our deep IC framework. For all methods, we use the same number of iterations, training loss and learning rate as used for our method.

- **DeepLK-6DoF**: We implemented a variant of DeepLK [111] which predicts the 3D transformation $\boldsymbol{\xi} \in \mathfrak{se}(3)$ instead of translation and scale prediction in their original 2D task. We use Gauss-Newton as the default optimization for this approach and no Convolutional M-Estimator. A comparison of this approach with our method when using only the two-view feature network (A) shows the benefits of our two-view feature encoder.

- **IC-FC-LS-Net**: We also implemented LS-Net [118] within our IC framework with the following differences to the original paper. First, we do not estimate or refine depth. Second, we do not use a separate network to provide an initial pose estimation. Third, we replace their LSTM layers with three fully connected (FC) layers which take the flattened $\mathbf{J}^T\mathbf{W}\mathbf{J}$ and $\mathbf{J}^T\mathbf{W}\mathbf{r}_k$ as input.

Ablation Study: We use (A), (B), (C) to refer to our contributions in Sec.6.1. We set $\mathbf{W}$ to the identity matrix when the Convolutional M-Estimator (B) is not used and use Gauss-Newton optimization in the absence of the Trust Region Network (C). We consider the following configurations:

- **Ours (A)+(B)+(C):** Our proposed method with shared weights. We perform coarse-to-fine iterations on four pyramid levels with three IC iterations at each level. We use shared weights for all iterations in (B) and (C).

- **Ours (A)+(B)+(C) (No WS):** A version of our method without shared weights. All settings are the same as above except that the network for (B) and (C) have independent weight parameters at each coarse-to-fine scale.

- **Ours (A)+(B)+(C) (K iterations/scale):** The same network as the default setting with shared weights, except that we change the inner iteration number K.

- **No Learning:** Vanilla coarse-to-fine IC alignment minimizing photometric error (6.7) without learned modules.

Table 6.1: **Quantitative Evaluation on MovingObjects3D.** We evaluate the average 3D EPE, angular error Θ, translation error $\mathbf{t}$ and success ratios $\mathbf{t} < 5\&\Theta < 5^\circ$ for three different motion magnitudes {Small, Medium, Large} which correspond to frames sampled from the original videos using frame intervals $\{1, 2, 4\}$.

	Model Descriptions	3D EPE (cm) ↓ on Validation/Test		
		Small	Medium	Large
ICP	Point-Plane ICP [147]	4.88/4.28	10.13/8.74	20.24/17.43
	Point-Point ICP [148]	5.02/4.38	10.33/9.06	20.43/17.68
	Oracle Point-Plane ICP [147]	3.91/3.31	10.68/9.63	22.53/19.98
	Oracle Point-Point ICP [148]	4.34/3.99	11.83/10.29	21.27/26.13
DPR	PoseCNN	5.18/4.60	10.43/9.20	20.08/17.74
	IC-PoseCNN	5.14/4.56	10.40/9.13	19.80/17.31
	Cascaded-PoseCNN	5.24/4.68	10.43/9.21	20.32/17.30
Learning Optim	No learning	11.66/11.26	21.85/22.95	37.01/38.88
	IC-FC-LS-Net, adapted from [118]	4.96/4.62	10.49/9.21	20.31/17.34
	DeepLK-6DoF, adapted from [111]	4.41/3.75	9.05/7.54	18.46/15.33
	Ours: (A)	4.35/3.66	8.80/7.23	18.28/15.06
	Ours: (A)+(B)	4.33/3.26	8.84/7.30	18.14/15.04
	Ours: (A)+(B)+(C)	**3.58**/2.91	**7.30/5.94**	**15.48/12.96**
	Ours: (A)+(B)+(C) (No WS)	3.62/**2.89**	7.54/6.08	16.00/12.98
	Ours: (A)+(B)+(C) ($K = 1$)	4.12/3.37	8.64/7.08	17.67/14.92
	Ours: (A)+(B)+(C) ($K = 5$)	3.60/2.92	7.49/6.09	16.06/13.01

6.5.3 Results and Discussion

Table 6.1 and Table 6.2 summarize our main results. For each dataset, we evaluate the method separately for three different motion magnitudes {Small, Medium, Large}. In Table 6.1, {Small, Medium, Large} correspond to frames sampled from the original videos at intervals $\{1, 2, 4\}$. In Table 6.2, [Small, Medium, Large] correspond to frame intervals $\{2, 4, 8\}$ on BundleFusion and $\{1, 2, 5\}$ on DynamicBundleFusion. We show the following metrics/statistics:

- **3D End-Point-Error (3D EPE):** This metric is defined in (6.21). We only evaluate errors on the rigidly moving objects, i.e., the moving objects in MovingObjects3D and the rigid background mask in DynamicBundleFusion.

Table 6.2: **Quantitative Evaluation on BundleFusion and DynamicBundleFusion.** In BundleFusion [95], the motion magnitudes {Small, Medium, Large} correspond to frame intervals $\{2, 4, 8\}$. In DynamicBundleFusion [146], the motion magnitudes {Small, Medium, Large} correspond to frame intervals $\{1, 2, 5\}$ (we reduce the intervals due to the increased difficulty).

	Model Descriptions	3D EPE (cm) ↓ Test on BundleFusion[95]			3D EPE (cm) ↓ Test on DynamicBundleFusion [146]		
		Small	Medium	Large	Small	Medium	Large
ICP	Point-Plane ICP [147]	2.01	4.52	11.11	0.97	2.09	4.89
	Point-Point ICP [148]	2.48	5.72	12.58	1.08	2.42	7.43
DPR	PoseCNN	3.41	6.85	13.31	1.65	3.19	8.24
	IC-PoseCNN	3.26	6.52	12.81	1.66	3.18	8.05
	Cascaded-PoseCNN	3.41	6.81	13.02	1.60	3.15	8.15
Learning Optim	No learning	3.35	6.30	12.51	3.39	4.69	8.58
	IC-FC-LS-Net, adapted from [118]	3.03	6.85	13.32	1.80	3.45	8.35
	DeepLK-6DoF, adapted from [111]	2.99	5.84	12.27	1.72	3.12	7.22
	Ours: (A)	2.65	5.46	11.92	1.65	2.96	7.11
	Ours: (A)+(B)	1.75	3.47	8.40	1.70	2.97	6.88
	Ours: (A)+(B)+(C)	**1.48**	**3.09**	7.84	0.74	1.54	4.64
	Ours: (A)+(B)+(C) (No WS)	1.52	3.10	**7.81**	**0.61**	**1.32**	**3.82**

- **Object rotation and translation:** We evaluate 3D rotation using the norm of Euler angles Θ, translation $\mathbf{t}$ in cm and the success ratio ($\mathbf{t} < 5$ (cm) & $\Theta < 5^\circ$), for the task of object motion estimation in Table 6.1.

- **Relative Pose Error**: We follow the standard visual odometry metric using relative Axis angle θ and translation $\mathbf{t}$ in cm.

- **Model Weight:** The number of learnable parameters.

- **Inference speed:** The forward execution time for a single image-pair at inference time, using GTX 1080 Ti.

3D Alignment Evaluation: Compared to all baseline methods (Table 6.1 row 1-10 and Table 6.2 row 1-8), our full model ((A)+(B)+(C)) achieves the overall best performance across different motion magnitudes and datasets while maintaining fast inference speed. Compared to ICP methods (ICP rows in Table 6.1 and Table 6.2) and classical method (No

	mRPE: θ (Deg) $\downarrow$ / t (cm) $\downarrow$				3D EPE (cm)			
	KF 1	KF 2	KF 4	KF 8	KF 1	KF 2	KF 4	KF 8
fr1/360								
RGBD VO [150]	0.46/1.03	2.45/5.26	7.47/10.31	16.08/17.32	1.33	5.98	21.34	52.50
Ours: (A)	0.50/1.33	1.32/3.84	5.68/11.79	14.33/16.26	1.24	2.39	13.37	49.60
Ours: (A)+(B)	0.45/1.18	1.00/3.18	4.96/11.62	14.23/17.52	1.18	1.92	10.67	49.13
Ours: (A)+(B)+(C)	**0.33/0.61**	**0.49/1.20**	**2.64/2.63**	**7.24/6.64**	**1.05**	**1.21**	**2.64**	**22.40**
fr1/desk								
RGBD VO [150]	**0.43/0.69**	**0.76/1.04**	3.52/5.15	10.71/19.83	**0.59**	**0.91**	5.46	18.84
Ours: (A)	0.58/1.04	1.12/2.05	2.75/4.87	7.14/11.27	0.74	1.31	3.89	12.89
Ours: (A)+(B)	0.57/1.02	1.08/2.03	2.54/4.63	6.59/10.87	0.74	1.28	3.42	11.21
Ours: (A)+(B)+(C)	0.55/0.90	0.89/1.55	**1.58/2.59**	**4.30/7.11**	0.68	0.96	**1.74**	**6.11**
fr2/desk								
RGBD VO [150]	0.30/0.56	**0.35/0.72**	0.79/1.78	1.44/3.80	0.96	**0.93**	2.20	4.75
Ours: (A)	0.30/0.54	0.44/0.98	0.72/1.94	1.46/4.30	0.80	1.04	1.60	2.93
Ours: (A)+(B)	0.31/0.56	0.45/1.03	0.77/2.04	1.43/4.14	0.80	1.04	1.62	2.83
Ours: (A)+(B)+(C)	**0.27/0.42**	0.39/0.73	**0.61/1.37**	**1.11/2.79**	**0.73**	0.94	**1.29**	**2.09**
fr2/pioneer_360								
RGBD VO [150]	1.02/1.82	2.00/4.19	4.20/6.51	8.54/14.36	6.39	9.22	21.16	48.46
Ours: (A)	0.76/1.77	1.04/3.66	2.34/8.16	7.60/14.83	3.55	5.67	13.78	39.41
Ours: (A)+(B)	0.72/1.80	0.95/3.52	2.15/6.96	6.91/13.31	3.57	5.42	12.77	36.80
Ours: (A)+(B)+(C)	**0.65/0.85**	**0.74/1.07**	**0.98/1.79**	**2.38/6.85**	**2.76**	**3.15**	**4.46**	**13.52**
Average over trajectories mRPE and EPE								
RGBD VO [150]	0.55/1.03	1.39/2.81	3.99/5.95	9.20/13.83	2.31	4.38	12.67	31.13
Ours: (A)	0.53/1.17	0.97/2.63	2.87/6.89	7.63/12.16	1.58	2.60	8.15	26.20
Ours: (A)+(B)	0.51/1.14	0.87/2.44	2.60/6.56	7.30/11.21	1.56	2.41	7.10	24.69
Ours: (A)+(B)+(C)	**0.45/0.69**	**0.63/1.14**	**1.10/2.09**	**3.76/5.88**	**1.31**	**1.57**	**2.53**	**11.03**
Average over frames mRPE and EPE								
RGBD VO [150]	0.48/0.90	1.08/2.20	2.95/4.60	6.54/10.26	1.80	3.53	9.58	23.14
Ours: (A)	0.46/0.98	0.79/2.12	2.16/5.35	5.58/9.65	1.39	2.18	6.22	19.16
Ours: (A)+(B)	0.45/0.96	0.73/2.00	1.99/5.15	5.35/9.41	1.38	2.05	5.52	18.31
Ours: (A)+(B)+(C)	**0.39/0.59**	**0.54/0.98**	**0.91/1.83**	**2.82/4.80**	**1.16**	**1.41**	**2.18**	**8.28**

Table 6.3: **Detailed Results on TUM RGB-D Dataset[104].** This table shows the mean relative pose error (mRPE) on our test split of the TUM RGB-D Dataset. KF denotes the size of the key frame intervals.

Learning in Table 6.1 and Table 6.2), our method achieves better performance without instance information at runtime. Besides, note that our model can achieve better performance with a significantly smaller number of weight parameters compared to direct image-to-pose regression (DRP rows in Table 6.1 and Table 6.2).

Visual Odometry Evaluation: To better understand our method operating under different conditions, we present a detailed quantitative evaluation in Table 6.3 using the Relative Pose Error (RPE) and the 3D End Point Error (3D EPE) metrics for each trajectory. Our method with all modules (A)+(B)+(C) outperforms the baseline RGBD visual odometry [150] across all subsampled trajectories, except in 'fr1/desk' of keyframe [1,2] and 'fr2/desk' of keyframe 2 in which the performance is close to each other. Our method shows clear advantages when the motion magnitude is large, e.g. 'fr1/360', 'fr2/pioneer_360'. From the ablation, we also observe that adding the trust-region module (C) helps to stabilize training and yields the most significant improvement in test accuracy. One possible reason for this is that the trust-region module can adjust the damping parameters and adapt to a wide range of motion magnitudes in the data. At training time, adaptive damping helps to stabilize the training loss and potentially contributes to learning better features in modules (A) and (B). At inference time, the method can adjust its trust-region step to adapt to different motion magnitudes with a fixed number of iterations.

Ablation Discussion: Across all ablation variants and datasets, our model achieves the best performance by combining all three proposed modules ((A)+(B)+(C)). This demonstrates that all components are relevant for robust learning-based optimization. Note that the influence of the proposed modules varies according to the properties of the specific task and dataset. For example, in the presence of inaccurate warping induced by noisy depth estimates from real data, learning a robust M-estimator (B) (Table 6.2 row 10 in BundleFusion) provides the most significant improvements. In the presence of heavy occlusions and motion ambiguities, learning the trust region step (C) helps to find better local optima which results in large improvements in our experiments, observed both when estimating

object motion (Table 6.1 row 13) and when estimating camera motion in dynamic scenes (Table 6.2 row 11 in DynamicBundleFusion). In complex dynamic environments, disabling weight sharing (Table 6.2 row 12 in DynamicBundleFusion) helps to capture context information and hence improves the results.

6.5.4 Qualitative Visualizations

We now demonstrate additional qualitative results on 3D rigid transformation estimation on MovingObjects3D. Note the difficulty of the task (truncation, independent background object) and the high quality of our alignments.

Visualizations of Iterative Estimation: Fig. 6.5 and Fig. 6.6 show a qualitative visualization of our method at different iterations. We warp the image $\mathbf{I}$ using the iterative estimated poses at the four coarse-to-fine pyramid scales ($\mathbf{I}(\boldsymbol{\xi}_1^\star) \rightarrow \mathbf{I}(\boldsymbol{\xi}_2^\star) \rightarrow \mathbf{I}(\boldsymbol{\xi}_3^\star) \rightarrow \mathbf{I}(\boldsymbol{\xi}_{\text{final}}^\star)$). $\mathbf{I}(\boldsymbol{\xi}_{\text{final}}^\star)$ is the final output of our method. Our results demonstrate that the proposed method is able to iteratively refine the estimation towards the global optimal solution despite the challenging scenario.

Comparison to Baselines: Fig. 6.7 shows a comparison of our full method to DeepLK 6DoF [111] and ablated models using only parts of the proposed modules. Our results demonstrate that the combination of all modules yields high-quality registrations.

6.6 2D Affine Motion Estimation

The proposed framework is general and can be applied to a wide range of motion models apart from the 3D rigid motion estimation tasks presented in the main paper, see, e.g., [129]. To demonstrate its generality, this section provides results on 2D affine motion estimation. While 2D affine motion estimation is in general easier than 3D rigid motion estimation, occlusions cannot be treated explicitly as depth is unknown. Thus, any successful method must implicitly identify occlusions as outliers during estimation.

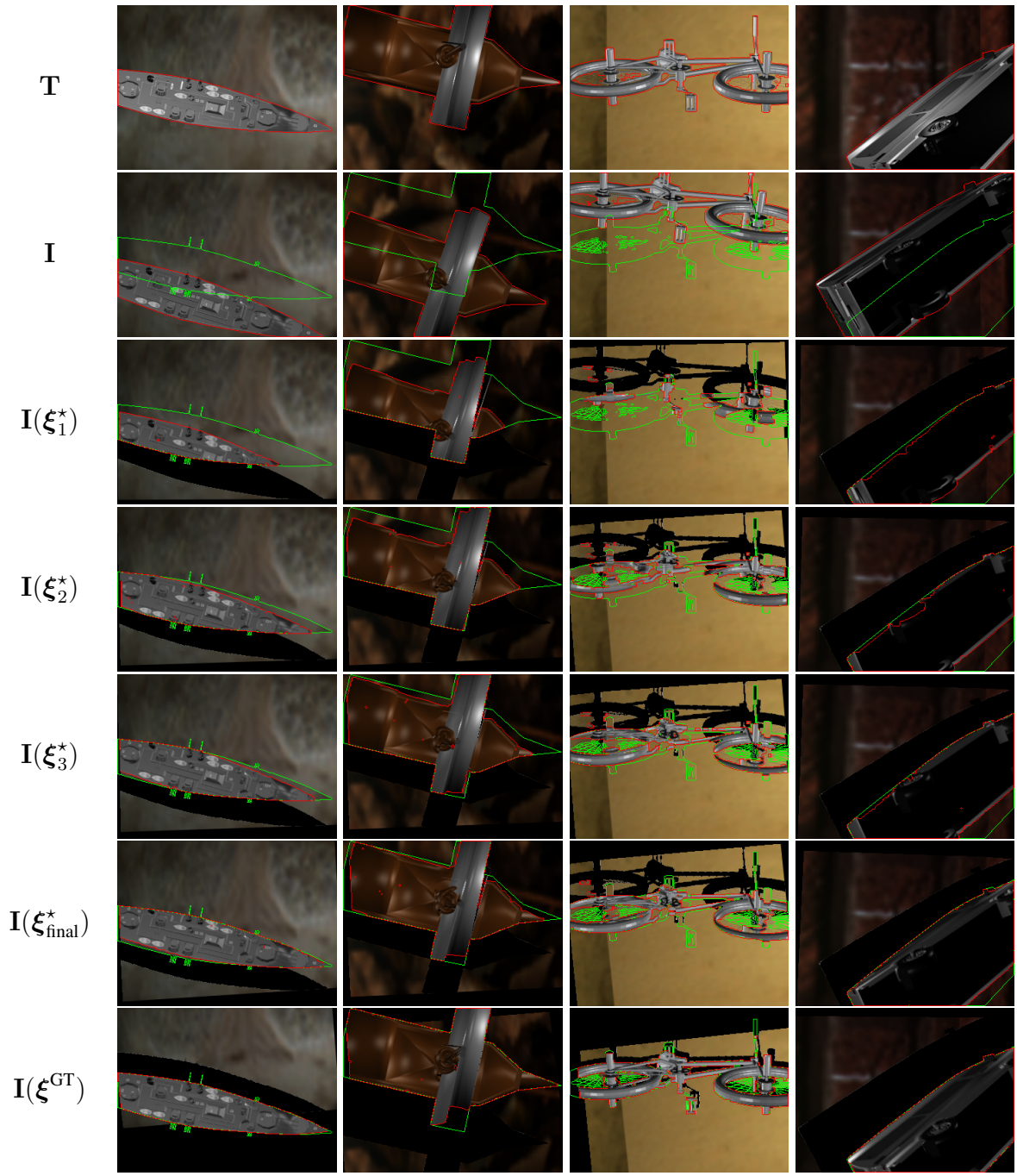


Figure 6.5: **Qualitative Results on MovingObjects3D.** Visualization of the warped image $\mathbf{I}(\boldsymbol{\xi})$ using the ground truth object motion $\boldsymbol{\xi}^{\text{GT}}$ (last row) and the object motion $\boldsymbol{\xi}^\star$ estimated using our method at each pyramid ($\boldsymbol{\xi}_1^\star$, $\boldsymbol{\xi}_2^\star$, $\boldsymbol{\xi}_3^\star$, $\boldsymbol{\xi}_{\text{final}}^\star$) on the MovingObjects3D *validation* and *test* set. In $\mathbf{I}$, we plot the instance boundary in red and that of $\mathbf{T}$ in green as comparison. Note the difficulty of the task (truncation, independent background object) and the high quality of our alignments. Black regions in the warped image are due to occlusion.

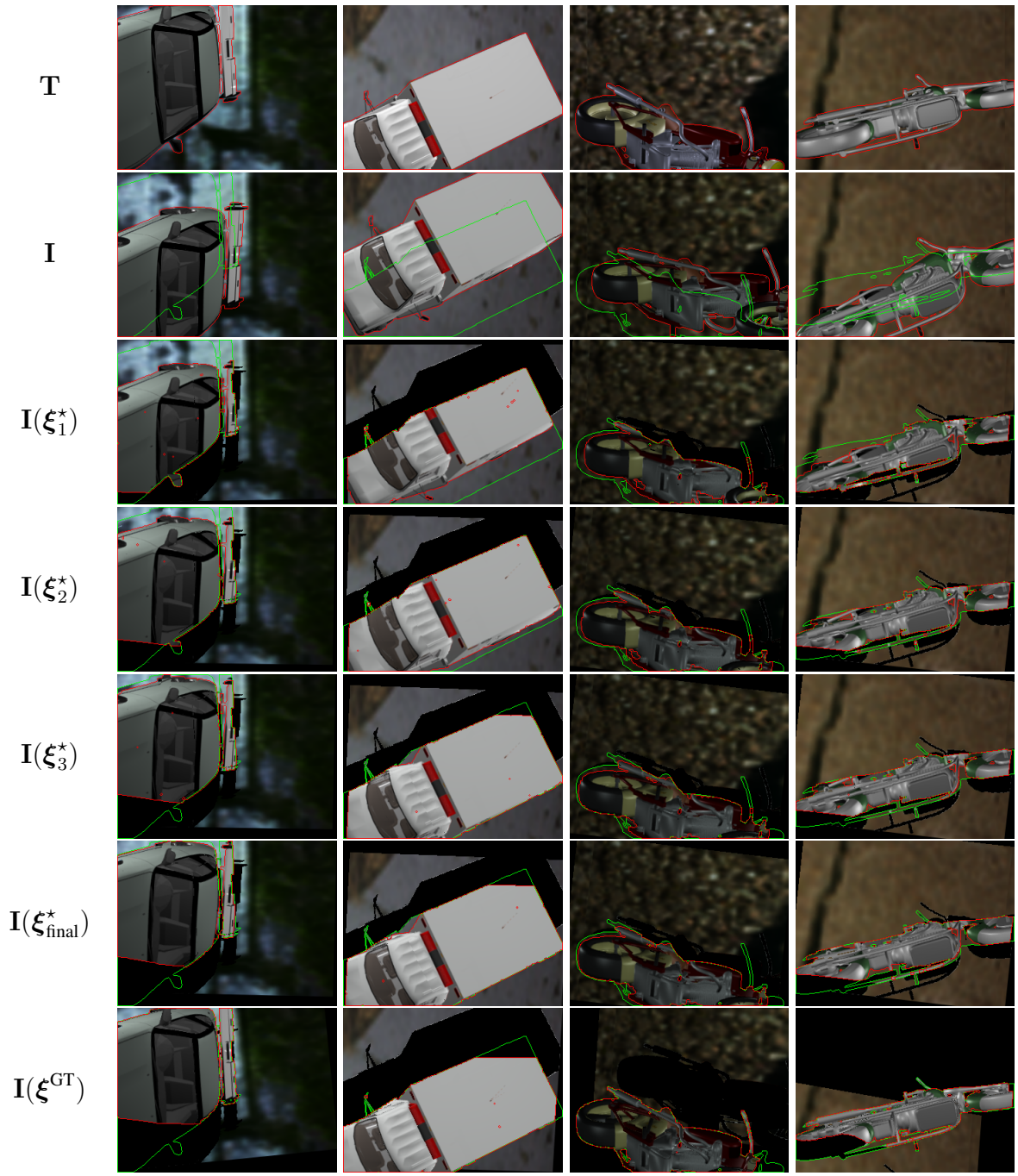


Figure 6.6: **Qualitative Results on MovingObjects3D.** Visualization of the warped image $\mathbf{I}(\boldsymbol{\xi})$ using the ground truth object motion $\boldsymbol{\xi}^{\text{GT}}$ (last row) and the object motion $\boldsymbol{\xi}^\star$ estimated using our method at each pyramid ($\boldsymbol{\xi}_1^\star$, $\boldsymbol{\xi}_2^\star$, $\boldsymbol{\xi}_3^\star$, $\boldsymbol{\xi}_{\text{final}}^\star$) on the MovingObjects3D *validation* and *test* set. In $\mathbf{I}$, we plot the instance boundary in red and that of $\mathbf{T}$ in green as comparison. Note the difficulty of the task (truncation, independent background object) and the high quality of our alignments. Black regions in the warped image are due to occlusion.

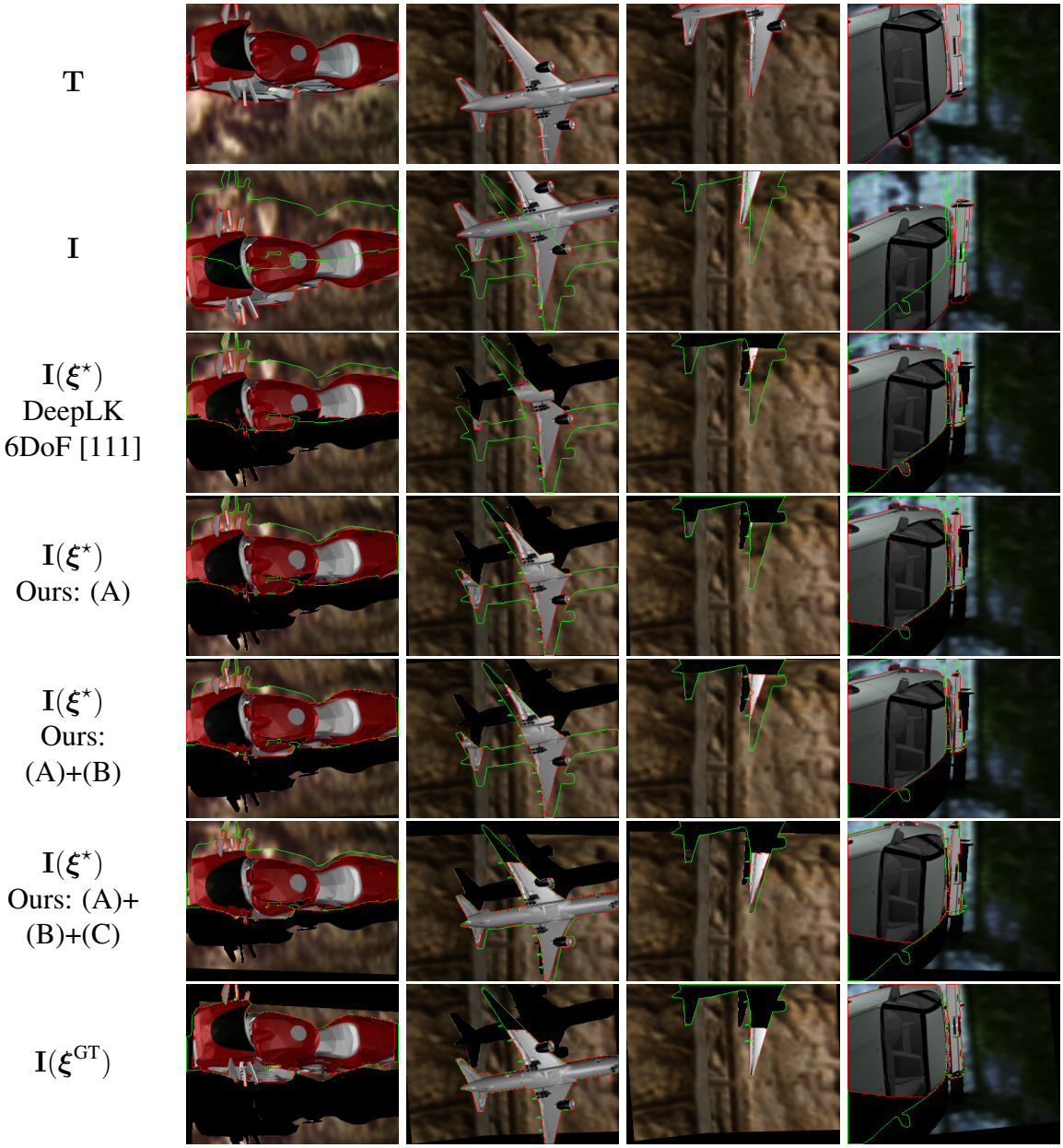


Figure 6.7: **Qualitative Comparisons of Our Method to Baselines on MovingObjects3D.** We compared the object motion $\boldsymbol{\xi}^{\star}$ estimated using our proposed modules (A)+(B)+(C) $\boldsymbol{\xi}^{\star}$ (row 6) to the optimal poses output from DeepLK 6DoF (row 3), ours (A) (row 4) and ours (A)+(B) (row 5) on the MovingObjects3D *validation* and *test* set. We visualize the warped image $\mathbf{I}(\boldsymbol{\xi})$ using the ground truth object motion $\boldsymbol{\xi}^{\text{GT}}$ (last row). In $\mathbf{I}$, we plot the instance boundary in red and that of $\mathbf{T}$ in green for qualitative comparison of the two shapes in 2D.

Implementation: Given pixel $\mathbf{x} = (x, y)^T \in \mathbb{R}^2$, we define the warping function $\mathbf{W}_{\boldsymbol{\xi}}$ using the following parameterization (see also [121])

$$\begin{bmatrix} 1+\xi_1 & \xi_3 & \xi_5 \\ \xi_2 & 1+\xi_4 & \xi_6 \end{bmatrix} \tag{6.22}$$

where $\boldsymbol{\xi} = (\xi_1, \xi_2, \xi_3, \xi_4, \xi_5, \xi_6)^T$. The analytic form of the Jacobian in (6.22) is

$$\begin{bmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{bmatrix} \tag{6.23}$$

The composition of $\boldsymbol{\xi} \circ \Delta\boldsymbol{\xi}$ is given by

$$\begin{bmatrix} \xi_1 + \Delta\xi_1 + \xi_1\Delta\xi_1 + \xi_3\Delta\xi_2 \\ \xi_2 + \Delta\xi_2 + \xi_2\Delta\xi_1 + \xi_4\Delta\xi_2 \\ \xi_3 + \Delta\xi_3 + \xi_1\Delta\xi_3 + \xi_3\Delta\xi_4 \\ \xi_4 + \Delta\xi_4 + \xi_2\Delta\xi_3 + \xi_4\Delta\xi_4 \\ \xi_5 + \Delta\xi_5 + \xi_1\Delta\xi_5 + \xi_3\Delta\xi_6 \\ \xi_6 + \Delta\xi_6 + \xi_2\Delta\xi_5 + \xi_4\Delta\xi_6 \end{bmatrix} \tag{6.24}$$

The parameters of the inverse affine ξ^{-1} in (6.22) are

$$\frac{1}{(1+\xi_1)(1+\xi_4) - \xi_2\xi_3} \begin{bmatrix} -\xi_1 - \xi_1\xi_4 + \xi_2\xi_3 \\ -\xi_2 \\ -\xi_3 \\ -\xi_4 - \xi_1\xi_4 + \xi_2\xi_3 \\ -\xi_5 - \xi_4\xi_5 + \xi_3\xi_6 \\ -\xi_6 - \xi_1\xi_6 + \xi_2\xi_5 \end{bmatrix} \tag{6.25}$$

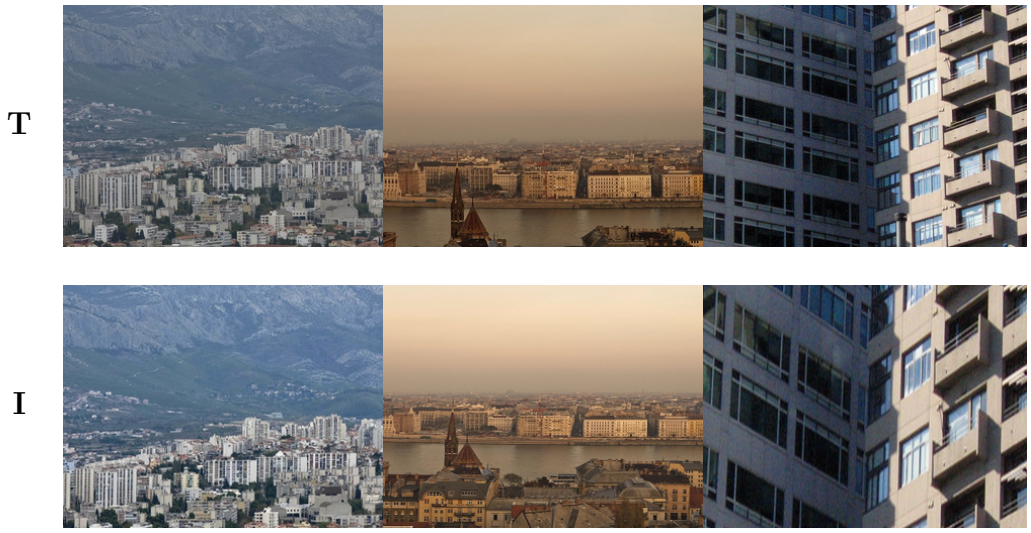


Figure 6.8: Examples of generated pairs $\mathbf{T}$ and $\mathbf{I}$ used for our 2D affine transformation estimation experiments.

6.6.1 Datasets Description

We download 1800 natural images from flickr and use them to generate a 2D dataset for training affine 2D transformation estimation. We use the downloaded flickr images as $\mathbf{T}$. Given a random affine transform which induces a 2D warping field, we synthesize the template $\mathbf{I}$ by applying the warping field to $\mathbf{T}$ using bilinear interpolation. To remove the boundary effects caused by zero padding in the warping process, we crop a region of size 240x320 from the central region of both $\mathbf{I}$ and $\mathbf{T}$. According to the actual raw image size and the coordinates of the cropped region, we adjust the warping field to ensure the cropped region has the correct affine transform. Fig. 6.8 shows generated pairs $\mathbf{T}$ and $\mathbf{I}$ examples.

For each template image, we randomly generate six different affine transforms, and synthesize six different images $\mathbf{I}$ using the warping field induced from each affine transform. The generated affine transforms are used as ground truth for training and evaluation. We use five out of the six generated affine transforms and their corresponding image pairs as the training set, and use the rest for testing.

	L1 error
No Learning	0.219
DeepLK, adapted from [111]	0.158
Ours: (A)	0.151
Ours: (A)+(B)	0.132
Ours: (A)+(B)+(C)	0.071

Table 6.4: Quantitative evaluation in 2D affine transform on the test set using L1 error.

6.6.2 Baselines and Results

We use (A), (B), (C) to refer to our contributions in Sec.1. We set $\mathbf{W}$ to the identity matrix when the Convolutional M-Estimator (B) is not used and use Gauss-Newton optimization in the absence of the Trust Region Network (C). We consider the following configurations:

- **Ours (A)+(B)+(C):** Our proposed method with shared weights. We perform coarse-to-fine iterations on three pyramid levels with three IC iterations at each level. We use shared weights for all iterations in (B) and (C).
- **DeepLK-6DoF:** We implemented a variant of DeepLK [111] which predicts the affine transform instead of translation and scale as in the original 2D task. We use Gauss-Newton as the default optimization for this approach and no Convolutional M-Estimator. Comparing this approach with our method when using only the two-view feature network (A) demonstrates the utility of our two-view feature encoder.

Training: During training, we minimizes the L1 norm of the distance from our estimated affine transform $\boldsymbol{\xi}^{\star}$ to the ground truth affine transform $\boldsymbol{\xi}^{\mathrm{GT}}$. We train our method and all baselines using a learning rate of 0.005.

Results: Table 6.4 shows a quantitative evaluation of our 2D affine motion estimation experiments. We evaluate the error in L1 norm by comparing the estimated results to the ground truth. Compared to all baseline methods, our model ((A)+(B)+(C)) yields the most accurate solution.

Note that different from the 3D motion estimation experiments in the main paper, there exists no motion ambiguity in the datasets used for 2D affine transform, rendering this setting simpler. We observe small improvements when using our two-view feature encoder (A) compared to using only a single view [111]. Using (B) Convolutional M-estimator gives better performance which may potentially address outliers induced by occlusions. Using (C) Trust Region Network further helps the network to boost performance. Similarly to our results in the main paper, we observe that we obtain the most accurate results when combining all components ((A)+(B)+(C)) of our model.

6.7 Conclusion

We have taken a deeper look at the inverse compositional algorithm by rephrasing it as a neural network with three trainable submodules which allows for relaxing some of the most significant restrictions of the original formulation. Experiments on the challenging task of relative rigid motion estimation from two RGB-D frames demonstrate that our method achieves more accurate results compared to both classical IC algorithms as well as data hungry direct image-to-pose regression techniques.

Although our data-driven IC method can better handle challenges in large object motions, heavy occlusions and varying illuminations, solving those challenges in the wild remains a subject for future research. To extend the current method to real-world environments with complex entangled motions, possible future directions include exploring multiple motion hypotheses, multi-view constraints and various motion models which will capture a broader family of computer vision tasks.

CHAPTER 7

LEARNING TO RECOVER MONOCULAR DEPTH AND SCENE FLOW FROM RIGID MOVING SCENES

Summary

In this work, we propose an approach to recover depth and 3D motion of dynamic rigid moving scenes observed from two consecutive monocular views. Different from the existing approaches that learn to regress depth from a single image, we propose to encourage depth from motion-parallax using least-square optimization. To cope with the moving scenes that violate epipolar geometry, we decompose the dynamic scene into rigid moving segments using instance segmentation and jointly optimize the geometry motion consistency of each rigid moving instance. We propose a novel learning re-weighting method to learn and optimize the optimal depth value efficiently. In real-world KITTI dataset as well as results on unseen videos, we demonstrate that the method can accurately recover monocular depth and scene motions jointly from rigid moving scenes. Our results outperform existing state-of-the-art learning-based and optimization-based methods in monocular depth estimation as well as in multi-view dynamic reconstruction.

7.1 Introduction

Recovering depth and 3D motion from monocular views is a well-studied task, generally referred to as Structure-from-Motion (SfM). A large body of existing work focuses on studying static scenes from multiple viewpoints or dynamic scenes recorded from a static view, while very few work address simultaneously recovering depth, camera motion, and scene motion in dynamic scenes observed from a monocular view sequence.

Although humans can interpret the object geometry and motion as an ego-moving observer, it is a fundamental unresolved challenge for machine vision since structure and motions are naturally ambiguous from monocular views. The depth scale is unknown from perspective motion stereo. Multiple triangulation methods in classical 3D geometry can recover depth from moving cameras [92], but these methods do not apply to simultaneous moving objects and camera which violate epipolar constraints.

Previous multi-view geometry approaches jointly optimize the rigid scene segmentation and depth from triangulation as an optimization problem [68, 67]. However, existing energy-based methods depend on vulnerable assumptions about object grouping and depth ordering in the scene structure and inference to obtain a solution is generally slow.

The recent progress data-driven based approaches demonstrate promising directions to address the problem via tackling the subproblems such as learning monocular depth [151, 152], camera pose regression [57], scene motion as optical flow [48, 146] and jointly learning them with geometric consistency in a self-supervised manner [57, 60, 58, 59]. While the existing independent networks can achieve remarkable performance in handling the ill-posed problem using image-based regression, they do not enforce multi-view geometry for the predicted motion and depth at inference time.

In this work, we argue for the importance of both optimization and learning in depth and motion recovery. We focus on solving the dynamic scene that can be composed of rigid moving scenes via instance detection as existing approaches [12, 32, 61], which is

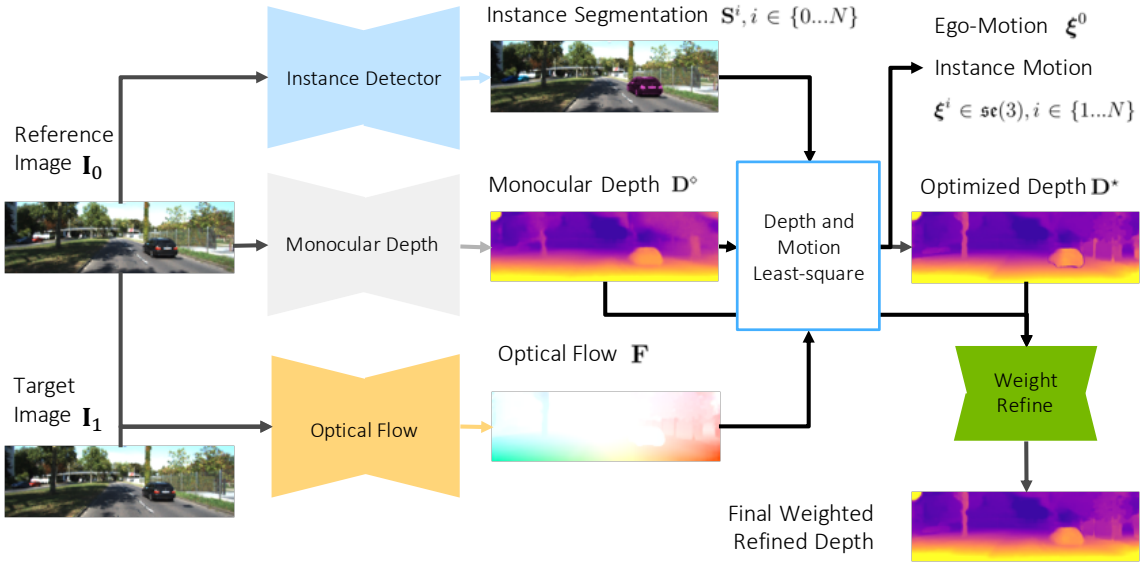


Figure 7.1: **An overview of our propose monocular instance scene flow method.** Our method takes two-view image as input. We first compute the instance segmentation $\mathbf{S}^i, i \in \{0..N\}$, a monocular depth $\mathbf{D}^\diamond$ on the reference image and an optical flow $\mathbf{F}$. With these information, we can recover the ego-motion $\boldsymbol{\xi}^0$, instance motion $\boldsymbol{\xi}^i, i \in \{1...N\}$ and depth as a least-square problem. To handle the effect of outliers, we train a refinement network predict the weights to linear blend the monocular depth and optimized depth.

common in the urban scenario. We predict the dense depth of the reference frame, 6 DoF motion for the moving camera, and each detected instance motion. First, we initialize our approach with the outputs from instance segmentation network, monocular depth network, and optical flow network. Second, we optimize the depth and 3D motions of each moving hypothesis by minimizing the geometric errors using iteratively Gauss-Newton approach. Finally, we learn a reweighted optimized depth to further smoothing and coping with outliers in optimization.

Specifically, we make the following key contributions:

- As far as we know, our approach is the first learnable approach that can simultaneously recover depth, 6DoF camera motion, and instance motions in a dynamic scenario. Our proposed method outperforms the state-of-the-art methods and can achieve good generalization.

- Our approach explicitly exploits motion parallax in the dynamic scene using joint depth and motion optimization.
- We propose a novel two-stage depth optimization and reweighting method that is fast to optimize and easy to learn with data.

7.2 Related Work

Scene Flow Estimation: Traditional approaches solve scene flow by jointly optimizing depth and motion with structured assumptions in the scene, which include variational optimization [153, 17, 26] or piece-wise moving slanted planes [90, 13, 15]. Incorporate instance recognition [12] or semantic segmentation [14] as additional information for scene flow optimization can better deal with textureless, reflective or fast-moving regions. Recently, Ma et al. [32] use an iterative Gauss-Newton optimization to estimate 3D instance motion from the binocular stereo, optical flow, and instance segmentation as input. All of the above scene flow methods takes requires calibrated stereo pairs, which can determine the absolute scale of depth. However, we focus on tackling scene flow estimation only from monocular pair of images, which is fundamentally more challenging due to the depth and motion ambiguity.

Monocular Reconstruction in Dynamic Scenes: Several monocular two-view dynamic reconstruction methods [154, 67, 68, 155] tackle a similar problem as ours. All of these work formulate it as an optimization problem and do not exploit learning. The optimization-based approach is to solve a complex inference problem with many unknowns, which requires structured assumptions about depth ordering and smoothness. The inference is thus slow and can not generalize well to new scenes without careful parameter tuning. In our work, we integrate learning into optimization. We use learned depth as initialization and learn to reweight the final optimized depth, which eases the optimization problem.

Learning Monocular Depth and Motion: Recently there is significant progress indi-

rectly learning 3D structure (depth) and motion from image(s), which includes pioneering work in monocular depth prediction [151, 156, 56], optical flow estimation [19, 48], and relative pose regression [57], which together can constitute a baseline for monocular depth and 3D motion recovery, Similar to us, Cao et al. . [61] also use object instances to factor dynamic scenes as individual moving instances. While these networks can achieve impressive results in each task either trained separately or jointly [57, 60, 157, 59, 58], these existing learning-based work do not leverage optimization to ensure geometric consistency at inference time.

We are not the first approach to explore learning depth from motion parallax and neither the first to combine learning and optimization. Ummenhofer et al. [54] use a cascaded depth and optical flow network to bootstrap depth from motion-parallax, which was then enhanced by [55]. Several approaches propose to learn an initial depth as monocular depth [118, 143, 158], virtual disparity[159] or latent depth code [160], and then fusing the multi-frame depth [158] or jointly optimize depth and camera motion using least-square update. [118, 143] further unrolls the least-square updates as differentiable layers. Although [160, 118, 143, 158, 159] are similar to us in enforcing motion-parallax through optimization, they only target a static environment by optimizing depth and camera motion. Our method extends this family of work to a dynamic environment. It is fundamentally more challenging in optimization and learning to recover depth and multiple instances of motion.

Dynamic environment breaks the epipolar geometry, which is the most fundamental challenge in geometry learning and optimization. Li et al. [161] leverage frozen humans videos to learn the depth for dynamic humans. They first learn a motion segmentation for human and triangulate the depth in the background region using optical flow. The final depth network output a refined depth map by fusing the triangulate depth and segmentation. However, they only focus on dynamic humans and do not tackle general rigid object motions, which are the primary focus of our approach.

7.3 Monocular Rigid Instance Scene Flow

In this section, we present our depth and 3D motion recovery approach. Our method takes two monocular view images $\mathbf{I}_0, \mathbf{I}_1 \in \mathbb{R}^{H \times W}$ as input where $\mathbf{I}_0$ is the reference frame. We assume the dynamic scene is composed of rigid instances which can be detected and segmented using instance segmentation network, e.g. Mask-RCNN [20]. $\mathbf{S}^i \in \mathbb{Z}$ is the segmentation mask for instance i detected from the reference frame $\mathbf{I}_0$. The target is to recover the camera ego-emotion $\boldsymbol{\xi}^0 \in \mathfrak{se}(3)$, each instance motion $\boldsymbol{\xi}^i \in \mathfrak{se}(3), i \in \{1...N\}$ for all N detected objects, and a pixel-pixel depth map $\mathbf{D}_0$ of the reference frame $\mathbf{I}_0$.

Our key idea is to exploit the motion-parallax to solve the depth and motion for each rigid instance from two consecutive views, which can be formulated as a least-squares optimization problem. The depth map can be initialized using sing-view depth estimation method [56, 162, 57], and motion parallax can be revealed from optical flow. Our method is composed of an initialization stage, optimization stage and refinement stage as shown in Fig. 7.1. We will describe in this section as following.

7.3.1 Initialization Stage

Our method leverages the image-based regressed depth as an initialization, an optical flow network for dense image correspondences and an instance segmentation network to detect and segment the rigid instances.

Single-view depth network: Our monocular depth network is similar to the disparity network in [56]. The output is a virtual disparity image from a monocular view, from which we calculate the depth using the virtual baseline in training time. This network provides us the flexibility to train monocular depth directly using supervised information and self-supervised using a stereo pair of view without relying on the additional temporal association. The output is a single image-based depth $\mathbf{D}^{\diamond}$ for the reference image $\mathbf{I}_0$.

Optical flow network: We acquire the optical flow $\mathbf{F} \in \mathbb{R}^2$ from the PWC-net [48]

which is trained supervised. Several multi-frame variants of PWC-net can also be trained in self-supervised manner using raw videos [163, 164].

Instance segmentation: We use Mask-RCNN network [20] with Res-Net 50 backbone pre-trained on MS-COCO dataset as a general instance detector. We only select the categories of pedestrians and vehicles as active class labels. We observe that the instance detector has a good generalization for large visible objects in unseen videos across different domains. We treat each of the detected as potential moving object and categorize their motion status as part of the optimization stage.

Our approach is not limited to the initialized depth, flow, and detection networks can be substituted to the proposed backbones with(out) pre-training. Our proposed method can also benefit from a potential better initialization as a result of the rapid research progress in each of the task. In Section 7.5, we will discuss our network training setting in detail to fairly compared to the existing methods.

7.3.2 Optimization Stage

Given the initial single image depth of the reference frame $\mathbf{D}^{\diamond}$, two-view optical flow $\mathbf{F}$, the mask of background $\mathbf{S}^0$ and all instances $\mathbf{S}^i, i \in \{1..N\}$, our objective is to estimate the optimal 3D motion $\boldsymbol{\xi}^{i,\star}$ and the optimal depth $\mathbf{D}^{\star}(\mathbf{u}, \mathbf{S}^i)$ for all pixels $\mathbf{u}$ corresponding to an instance segment $\mathbf{S}^i, i \in \{0...N\}$. To simplify the notation in the following, we use $\mathbf{D}^{i,*}$ short for $\mathbf{D}^{\star}(\mathbf{u}, \mathbf{S}^i)$. The robust least-squares objective for an instance i is thus

$$\mathbf{D}^{i,\star}, \boldsymbol{\xi}^{i,\star} = \underset{\mathbf{D}^i, \boldsymbol{\xi}^i}{\operatorname{argmin}} \sum_{\mathbf{u}, \mathbf{S}^i} \rho\Big(\mathbf{r}(\boldsymbol{\xi}^i, \mathbf{D}^i(\mathbf{u}); \mathbf{F}(\mathbf{u}))\Big) \tag{7.1}$$

where $\rho(\cdot)$ is a robust Cauchy M-estimator defined in Section 3.2.3 and $\mathbf{r}$ is the residual function for pixel $\mathbf{u}$. Assuming each instance has independent motion, we solve the motion $\xi^{i,\star}$ and depth $\mathbf{D}^{i,\star}$ corresponding to segmentation mask $\mathbf{S}^i$ sequentially.

Residual function: For every pixel $\mathbf{u}$, we can obtain the rigid flow $\mathbf{F}^i_{rigid}(\mathbf{u}) \in \mathbb{R}^2$ by

projecting its corresponding the depth $\mathbf{D}^i(\mathbf{u})$ into the next view using the estimated motion $\boldsymbol{\xi}^i \in \mathfrak{se}(3)$ for segment $\mathbf{S}^i$. By comparing $\mathbf{F}^i_{rigid}$ to optical flow $\mathbf{F}$, we calculate the per pixel geometric residual $r^i(\boldsymbol{\xi}, D(\mathbf{u}))$ as

$$\mathbf{r}(\boldsymbol{\xi}^i, \mathbf{D}^i(\mathbf{u})) = \mathbf{F}^i_{rigid}(\mathbf{u}; \boldsymbol{\xi}^i, \mathbf{D}^i(\mathbf{u})) - \mathbf{F}(\mathbf{u}) \tag{7.2}$$

$$\mathbf{F}^i_{rigid}(\mathbf{u}; \boldsymbol{\xi}^i, \mathbf{D}^i(\mathbf{u})) = \mathcal{W}_\pi\big(\mathbf{T}(\boldsymbol{\xi})\pi^{-1}(\mathbf{u}, \mathbf{D}(\mathbf{u}))\big) - \mathbf{u} \tag{7.3}$$

where $\mathcal{W}_\pi(\cdot) : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ is the perspective image warping function defined in (6.20) and $\pi^{-1} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ is the inverse projection in (3.3), given the depth at pixel $\mathbf{D}(\mathbf{u})$. Calculating the Jacobian of the residual function w.r.t the depth $\mathbf{J}^i_{\mathbf{D}}(\mathbf{u})$ can refer to (3.27) and the Jacobian w.r.t the 3D motion $\mathbf{J}^i_{\boldsymbol{\xi}}(\mathbf{u})$ can refer to (3.28).

Moving object detection: In most of the frames, not all N instances recognized by the instance detector are moving instances. Although our algorithm can compute a motion hypothesis for each of these *static* objects, detecting their motion status can help us to skip the static objects and simplify the computation. We first calculate the ego-motion $\boldsymbol{\xi}^0 \in \mathfrak{se}(3)$ and the resulting ego-motion rigid flow as $\mathbf{F}^0_{rigid} \in \mathbb{R}^{2\times H\times W}$. For all detected instances i, we compute the instance flow residual w.r.t the rigid flow as $\mathbf{r}^i_{rigid}$ in (7.2).. If the median of the rigid flow magnitude $median(||\mathbf{F}^i(\mathbf{u}))|| < 1$ or the median of flow residual magnitude $median(||\mathbf{r}^i_{rigid}(\mathbf{u}))|| < 0.01$, we mark the instance as static object and assign the background motion hypothesis to it.

Efficient optimization: For every frame, the total number of unknowns are N 6DoF motion variables $\boldsymbol{\xi}^i \in \mathfrak{se}(3), i \in \{0...N\}$ and per-pixel depth $\mathbf{D} \in \mathbb{R}^{H\times W}$. To efficiently perform the optimization, we make the following assumptions to simplify the computation. First, we assume every moving object is independent. Second, we assume every depth value is also independent of its neighborhood. This first assumption reduces the joint hessian matrix to a block diagonal matrix so that we can solve each moving object motion $\boldsymbol{\xi}^i$ sequentially. The second assumption reduces the Hessian block of the depth matrix as a

Figure 7.2: **A visualization of the moving object detection.** Given the instance segmentation (visualized at top row), we check the motion status of the instance and ignore all the static instances. The segmentation of final moving objects are visualized at the bottom row.

diagonal matrix. The inverse of the diagonal matrix can be calculated in parallel efficiently. Solving the inverse block-diagonal matrix can refer to Schur complement. To further simplify the process, we solve the least-square optimization of motion and depth variables in an alternative way. We observe that it can achieve faster inference than blockwise elimination using Schur complement with approximated similar results.

With the above assumptions, we can solve the robust least-square objective in (7.1) very efficiently via the iterative Gauss-Newton optimization. In each iteration, we perform the Gauss-Newton update to the unknown variable $\boldsymbol{\xi}$ and $\mathbf{D}$ alternatively. In the (k) iteration, for the detected instance i (background when $i = 0$), the Gauss-Newton updates for an instance $\boldsymbol{\xi}^i_{(k)}$ and its corresponding depth $\mathbf{D}^i_{(k)}$ are

$$\delta\boldsymbol{\xi}^i_{(k)} = -(\mathbf{J}^T_{\boldsymbol{\xi}}\mathbf{W}_{(k)}\mathbf{J}_{\boldsymbol{\xi}})^{-1}\mathbf{J}^T_{\boldsymbol{\xi}}\mathbf{W}_{(k)}\,\mathbf{r}(\boldsymbol{\xi}^i, \mathbf{D}^{i,(k-1)}) \tag{7.4}$$

$$\boldsymbol{\xi}^i_{(k)} = \boldsymbol{\xi}^i_{(k-1)} \circ \delta\boldsymbol{\xi}^i_{(k)} \tag{7.5}$$

$$\delta\mathbf{D}^i_{(k)} = -(\mathbf{J}^T_{\mathbf{D}}\mathbf{W}_{(k)}\mathbf{J}_{\mathbf{D}})^{-1}\mathbf{J}^T_{\mathbf{D}}\mathbf{W}_{(k)}\,\mathbf{r}(\boldsymbol{\xi}^i_{(k)}, \mathbf{D}^i_{(k-1)}) \tag{7.6}$$

$$\mathbf{D}^i_{(k)} = \mathbf{D}^i_{(k-1)} + \delta\mathbf{D}^i_k \tag{7.7}$$

where $\mathbf{J}_{\mathbf{D}}$ and $\mathbf{J}_{\boldsymbol{\xi}}$ are the Jacobian functions for the two variables and $\mathbf{W}$ is the weight matrix derived from robust mestimator depending on the residual $\mathbf{r}_{(k)}$. Algorithm 5 describes

our optimization process.

Algorithm 5: The iterative Gauss-Newton updates for 3D motion and depth recovery.

// We perform a total of $K = 3$ iterations

for $k < K$ **do**

  // Update the rigid background motion

  $\delta\boldsymbol{\xi}^0_{(k)} = -(\mathbf{J}^T_{\boldsymbol{\xi}}\mathbf{W}_{(0)}\mathbf{J}_{\boldsymbol{\xi}})^{-1}\mathbf{J}^T_{\boldsymbol{\xi}}\mathbf{W}_{(k)}\,\mathbf{r}(\boldsymbol{\xi}^0_{(k-1)}, \mathbf{D}^0_{(k-1)})$;

  $\boldsymbol{\xi}^0_{(k)} = \boldsymbol{\xi}^0_{(k-1)} \circ \delta\boldsymbol{\xi}^0_{(k)}$;

  // Update the rigid background depth

  $\delta\mathbf{D}^0_{(k)} = -(\mathbf{J}^T_{\mathbf{D}}\mathbf{W}_{(k)}\mathbf{J}_{\mathbf{D}})^{-1}\mathbf{J}^T_{\mathbf{D}}\mathbf{W}_{(k)}\,\mathbf{r}(\boldsymbol{\xi}^0_{(k)}, \mathbf{D}^0_{(k)})$;

  $\mathbf{D}^0_k = \mathbf{D}^0_{(k-1)} + \delta\mathbf{D}^0_k$;

  // Update instance i motion and depth

  for $i \in \{1...N\}$ **do**

    $\mathbf{r}^i_{rigid} = \mathbf{F}^i(\boldsymbol{\xi}^0_{(k)}, \mathbf{D}^0_{(k-1)}) - \mathbf{F}^i$;

    if $median(||\mathbf{F}^i(\mathbf{u}))|| < 1$ *or* $median(||\mathbf{r}^i_{rigid}(\mathbf{u}))|| < 0.01$, $\mathbf{u} \in \mathbf{S}^i$ **then**

      $\delta\boldsymbol{\xi}^i_{(k)} = -(\mathbf{J}^T_{\boldsymbol{\xi}}\mathbf{W}_{(k)}\mathbf{J}_{\boldsymbol{\xi}})^{-1}\mathbf{J}^T_{\boldsymbol{\xi}}\mathbf{W}_{(k)}\,\mathbf{r}(\boldsymbol{\xi}^i, \mathbf{D}^{i,(k-1)})$;

      $\boldsymbol{\xi}^i_{(k)} = \boldsymbol{\xi}^i_{(k-1)} \circ \delta\boldsymbol{\xi}^i_{(k)}$;

    end

    else

      $\boldsymbol{\xi}^i_{(k)} = \boldsymbol{\xi}^0_{(k)}$;

    end

    $\delta\mathbf{D}^i_{(k)} = -(\mathbf{J}^T_{\mathbf{D}}\mathbf{W}_{(k)}\mathbf{J}_{\mathbf{D}})^{-1}\mathbf{J}^T_{\mathbf{D}}\mathbf{W}_{(k)}\,\mathbf{r}(\boldsymbol{\xi}^i_{(k)}, \mathbf{D}^i_{(k-1)})$;

    $\mathbf{D}^i_k = \mathbf{D}^i_{(k-1)} + \delta\mathbf{D}^i_k$;

  end

end

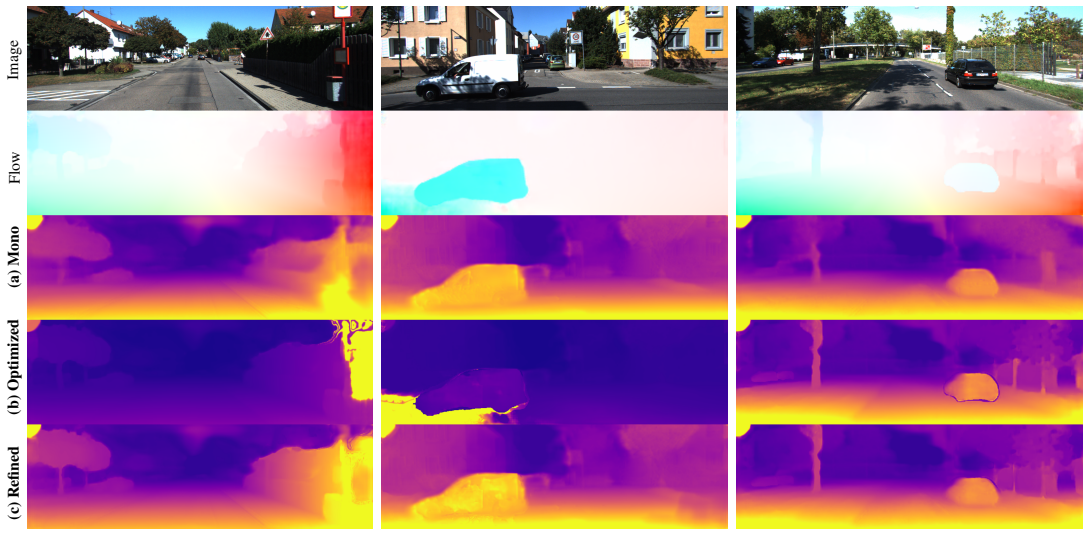


Figure 7.3: **A visualization of the challenges in depth estimation.** Compared to (a) initial monocular output, the depth optimization output (b) causes some artifacts in regions where estimated optical flow are noisy or wrong (left & center images) in optical flow, or mask is inconsistent with the flow in boundary (right). By learning (c) refined weighted depth map, we can fix these issues and preserve the accurate geometry structure.

For pose optimization in (7.5), there are extensive flow measurements in one segmentation mask to solve a single instance motion and the outliers in the measurements can be handled well via robust m-estimator. By assuming no neighborhood smoothness, we solve (7.6) independently for all pixels using its corresponding flow vector.

7.3.3 Refinement Stage: Learning Reweighted Depth

The challenges for the depth optimization: However, our depth optimization does not consider the smoothness of the depth pixels. The value of the depth at a pixel only depends on its initial depth estimation and the current projected residual, which makes the optimization very sensitive to outliers caused by the monocular depth and flow correspondence. The outliers in the residuals will significantly impact the optimized depth, primarily due to three sources of outliers. First, the depth optimization depends on the estimated value of its corresponding instance motion and optical flow. When the optical flow boundary and instance boundary does not match each other, the depth at the boundary pixels will be optimized by

a wrong association. Second, estimated optical flow is also impacted by occlusion, aperture problem, and potential domain shift. The errors of flow will also propagate to the depth estimation. Third, for static or distance scenes, the magnitude of flow vectors are noisy and small, which causes numerical instability. In this section, we will revisit the depth optimization problem and discuss how we can learn the outliers and smoothness for the optimization and address this challenge.

Cumulative Gauss-Newton update for depth: After all K iterations, the cumulative Gauss-Newton update (7.7) for the final optimized $\mathbf{D}^\star$ can be summarized as:

$$\mathbf{D}^\tau = \mathbf{D}^\diamond + \sum_{k=1:K} \delta\mathbf{D}_{(k)} \tag{7.8}$$

where we gather all the instances together and drop the superscript i.

Damped Cumulative Gauss-Newton update: The major issue in depth optimization is that a noisy measurement will cause a sensitive depth update and potentially deviate too much from the initial solution. One potential solution is to add a damping term to (7.7) that $\mathbf{D}^i_k = \mathbf{D}^i_{(k-1)} + \boldsymbol{\lambda}_{(k)}\delta\mathbf{D}^i_{(k)}$, where $\boldsymbol{\lambda}_k \in \mathbb{R}^{H\times W}$ is a per-pixel damping ranging from 0 to 1. As (7.8), we expect the optimal depth estimation $\mathbf{D}^\star$ can be obtained via a series of damped cumulative Gauss-Newton updates

$$\mathbf{D}^\star = \mathbf{D}^\diamond + \sum_{k=1:K} \boldsymbol{\lambda}_{(k)}\delta\mathbf{D}_{(k)} \tag{7.9}$$

$$= \mathbf{D}^\diamond + \boldsymbol{\lambda}_K \sum_{k=1:K} \delta\mathbf{D}_{(k)} \tag{7.10}$$

where we use a single damping term $\boldsymbol{\lambda}_K$ to subsume all the linear per step damping $\lambda_k, k \in \{1..K\}$ to simplify the damping estimation. $\boldsymbol{\lambda}_K$ is also in the range from 0 to 1. Substituting

$\sum_{k=1:K} \delta \mathbf{D}_{(k)}$ by $\mathbf{D}^{\tau} - \mathbf{D}^{\diamond}$ in (7.8), the optimal depth estimation $\mathbf{D}^{\star}$ is

$$\mathbf{D}^{\star} = \mathbf{D}^{\diamond} + \boldsymbol{\lambda}_K(\mathbf{D}^{\tau} - \mathbf{D}^{\diamond}) \tag{7.11}$$

$$= (\mathbf{I} - \boldsymbol{\lambda}_K)\mathbf{D}^{\diamond} + \boldsymbol{\lambda}_K \mathbf{D}^{\tau} \tag{7.12}$$

(7.12) shows the optimal damped solution can a linear composition of the initial depth estimation $\mathbf{D}^{\diamond}$ and the optimized depth using K Gauss-Newton updates $\mathbf{D}^{\tau}$. An optimal damping can potentially regularize the outliers and smoothness, which is unknown. With sufficient depth supervision, we can learn $\boldsymbol{\lambda}_K$ from data.

Depth Reweighting Network: We propose to learn the linear weighting using a convolutional neural network $\mathbf{W}_{\theta} = f_{\theta}(\mathbf{I}_0, \mathbf{D}^{\diamond}, \mathbf{D}^{\tau})$. The network inputs an initial single image monocular depth $\mathbf{D}^{\diamond}$, an optimized depth $\mathbf{D}^{\tau}$ as well as the reference image $\mathbf{I}_0$. It outputs a weight matrix $\mathbf{W}_{\theta} = \mathbf{I} - \boldsymbol{\lambda}_K, \mathbf{W}_{\theta} \in \mathbb{R}^{H \times W}$ in the range from 0 to 1. The final optimal depth $\mathbf{D}^{\star}$ is a reweighting the two depth map using $\mathbf{W}_{\theta}$

$$\mathbf{D}^{\star} = \mathbf{W}_{\theta}\mathbf{D}^{\diamond} + (\mathbf{I} - \mathbf{W}_{\theta})\mathbf{D}^{\tau} \tag{7.13}$$

During training, the network learns the spatial correlation of $\mathbf{D}^{\diamond}$ and $\mathbf{D}^{\tau}$. Empirically, we find the network can learn to smooth the depth estimation, handle the wrong optimized depth region due to occlusion and boundary inconsistency. Fig. 7.3 show a visualization example of several challenges discussed in Section 7.3.2. With the learned adaptive weighting, the final refined output can fix the optimization issues and meanwhile preserve the sharper and more accurate result benefiting from two-view parallax optimization.

Architecture: The Depth Reweighting Network is composed of five convolutional layer following by five transposed convolutional layers. The final layer is a sigmoid layer which normalizes the output to the range from 0 to 1.

7.4 Training Objectives

In the urban scenario, we can acquire the ground truth depth by projecting the sparse Lidar scans into reference images. We use the semi-supervised approach as [162] regressing the depth map to compensate the sparsity in the supervision signal. Since the ground truth of 3D motions is difficult to acquire, we do not use any supervision for motion regression.

Semi-supervised objectives: We view the training as a combination of supervised depth regression and a self-supervised view synthesis problem with the following objectives

$$\mathcal{L} = \underbrace{\alpha_{L1}\,\mathcal{L}_{L1} + \alpha_{sc}\,\mathcal{L}_{sc}}_{\text{supervised loss}} + \underbrace{\alpha_{sm}\,\mathcal{L}_{sm} + \alpha_{am}\,(\mathcal{L}^{l}_{am} + \mathcal{L}^{r}_{am})}_{\text{self-supervised loss}} \tag{7.14}$$

where $\mathcal{L}_{L1}$ is the inverse depth error in absolute scale, $\mathcal{L}_{sc}$ is the scale-invariant mean-square error, $\mathcal{L}_{sm}$ is the smoothness loss, α_{am} is the appearance matching term and $\mathcal{L}_{rc}$ is the reconstruction consistency term. We discuss the details of each term in the following.

L1 Depth Loss: We apply a point-wise L1 norm on the depth of all pixels which has valid projected depth from Lidar scan as

$$\mathcal{L}_{L1} = \frac{1}{N}\sum_{\mathbf{u}\in\Omega} \rho_{ch}(\mathbf{D}(\mathbf{u}) - \mathbf{D}^{gt}(\mathbf{u}))) \tag{7.15}$$

where ρ_{ch} is the Charbonnier function $\rho(x) = (x^2 + 1e^{-8})^{0.45}$, which can better cope with outliers during training. $\mathcal{L}_{L1}$ regresses the depth to the ground truth in absolute scale.

Scale-Invariant MSE: We use the scale-invariant mean square error (MSE) $\mathcal{L}_{sc}$ to enforce the relative scale estimation. $\mathcal{L}_{sc}$ computes the relative ratio of depth values and compare to the relative ratio of ground truth [151]

$$\mathcal{L}_{sc} = \frac{1}{N}\sum_{\mathbf{u}\in\Omega}\Big(\mathbf{D}(\mathbf{u}) - \mathbf{D}^{gt}(\mathbf{u})^2\Big) - \frac{1}{N^2}\Big(\sum_{\mathbf{u}\in\Omega}\mathbf{D}(\mathbf{u}) - \mathbf{D}^{gt}(\mathbf{u})\Big)^2 \tag{7.16}$$

Edge-aware Smoothness Loss: We incorporate the smoothness term $\mathcal{L}_{sm}$ to provide a smooth interpolation of depth in texture-less regions and regularize the depth estimation in which only partial pixels have ground truth supervision. The smoothness loss is weighted using first-order gradients ∇ and second-order gradients ∇^2 as

$$\mathcal{L}_{sm} = \frac{1}{N}\sum_{\mathbf{u}\in\Omega} |\nabla_x \mathbf{I}(\mathbf{u})| \exp(-||\nabla_x^2 \mathbf{I}(\mathbf{u})||) + |\nabla_y \mathbf{I}(\mathbf{u})| \exp(-||\nabla_y^2 \mathbf{I}(\mathbf{u})||) \tag{7.17}$$

Left-right Appearance Matching Loss: Our network predict a left-right virtual disparity from a single monocular image as [56]. During training, we similarly use a combination of L1 and single-scale SSIM term as the appearance matching loss

$$\mathcal{L}_{am}^{l} = \frac{1}{N}\sum_{\mathbf{u}\in\Omega} \alpha_{ssim}\frac{1}{2}(1 - \text{SSIM}(\mathbf{I}^l(\mathbf{u}), \mathbf{I}^r(\mathbf{u} + d^l))) + \tag{7.18}$$

$$(1 - \alpha_{ssim})||\mathbf{I}^l(\mathbf{u}) - \mathbf{I}^r(\mathbf{u} + d^l))|| \tag{7.19}$$

where $d^l \in \mathbb{R}$ is the disparity value corresponds to pixel $\mathbf{u}$ in the left reference image. Also following [56], we also use a simplified SSIM with a 3x3 block filter instead of a Gaussian, with $\alpha_{ssim} = 0.85$. During training, (7.18) is added bidirectional from left to right as $\mathcal{L}^l$ and right to left as $\mathcal{L}^r$.

7.5 Experiments

Implementation: We implement our approach all in Pytorch 1.0.0, including Gauss-Newton optimization. Although the approach is fully end-to-end differentiable, we do not observe additional performance gain to learn it end-to-end. Instead, we find it is significantly faster to train the method stage-by-stage. In the initialization stage, we train the monocular depth estimation, optical flow network independently. We use MS-COCO pre-

trained Mask-RCNN [1] to compute the instance segmentation for all of the experiments. In the refinement stage, we train the Depth Reweighting Network using the depth supervision with train (7.15), (7.16) and (7.17). For single-view monocular depth network and Depth Reweighting Network, we resize the input to 256x768 for both training and testing. In the optimization stage, we perform the pose Gauss-Newton at a reduced resolution of 128x384 and the depth Gauss-Newton update at 256x768.

Inference time: The initialization stage takes rougly 0.3s for extract monocular depth. The optimization takes 0.5s. The Depth Reweighting Network takes 0.01s. All for a single image of size 378x1275.

Training setting: Our method is not limited to a particular type of supervision. The method can fully leverage both raw stereo videos and depth from sparse supervision. In the following experiments, we train and evaluate our method in both settings.

- **Semi-supervised training:** The monocular depth network is trained with a mixed of all the training objectives in Section 7.4 following the KITTI raw dataset split. We use the PWC-net [48] pretrained on synthetic data and then finetuned on KITTI SceneFlow dataset using optical flow ground truth only.

- **Self-supervised training:** We train the monocular depth network with self-supervised losses in (7.18) and (7.17). We use the self-supervised trained optical flow method [163], which is a close variant of PWC-net.

Depth Evaluation Metric: For quantitative evaluation, we follow the standard monocular depth evaluation metric as Eigen et al. [151] as following

- **Threshold error δ (%):** $\max(d/d^{gt}, d^{gt}/d) < T$, where T is the threshold. We use the standard three thresholds $1.25, 1.25^2, 1.25^3$.

[1] https://github.com/facebookresearch/maskrcnn-benchmark

- **Absolute relative difference (abs_rel)**: $\frac{1}{n}\sum |d - d^{gt}|/d^{gt}$ This error is also termed as *mean relative error (MRE)* in [155, 68, 67].

- **Square relative different (sq_rel)**: $\frac{1}{n}\sum ||d - d^{gt}||^2/d^{gt}$

- **Relative mean square error (rmse):** $\sqrt{\frac{1}{N}\sum ||d - d^{gt}||}$

- **Relative mean square error in log scale (rmse_log):** $\sqrt{\frac{1}{N}\sum ||\log d - \log d^{gt}||}$

Among all of these metrics, abs_rel, sq_rel and δ are scale-invariant error, while rmse and rmse_log are scale dependent. Since depth and motion are tightly coupled, the quality of depth also indirectly demonstrate the motion accuracy.

7.5.1 Baseline and Ablations

Baselines: To fairly evaluate the task in jointly recovering depth and motion in the dynamic scenes, we compare our approach to the following two family of methods.

- **Joint learning depth and flow methods:** The recent progress in jointly monocular depth and motion estimation methods provide competitative baselines in learning monocular depth. These include approaches that jointly learn camera motion and depth by assuming the scene is quasi-static [57, 75, 157] or additionally infer non-rigid optical flow by assuming the scene is fully dynamic [60, 58, 59].

- **Dynamic reconstruction methods:** There are several dynamic reconstruction methods that recover the depth from two views [68, 68, 67] or multiple views [165, 166, 66]. All of these methods are optimization-based method and do not use learning.

Ablations: For monocular depth estimation, we also evaluate our intermediate results as ablation for our method termed as following.

- **Monocular-D:** Our method trained using a single-view monocular depth estimation method as [56], which is the result after the initialization stage in Section 7.3.1.

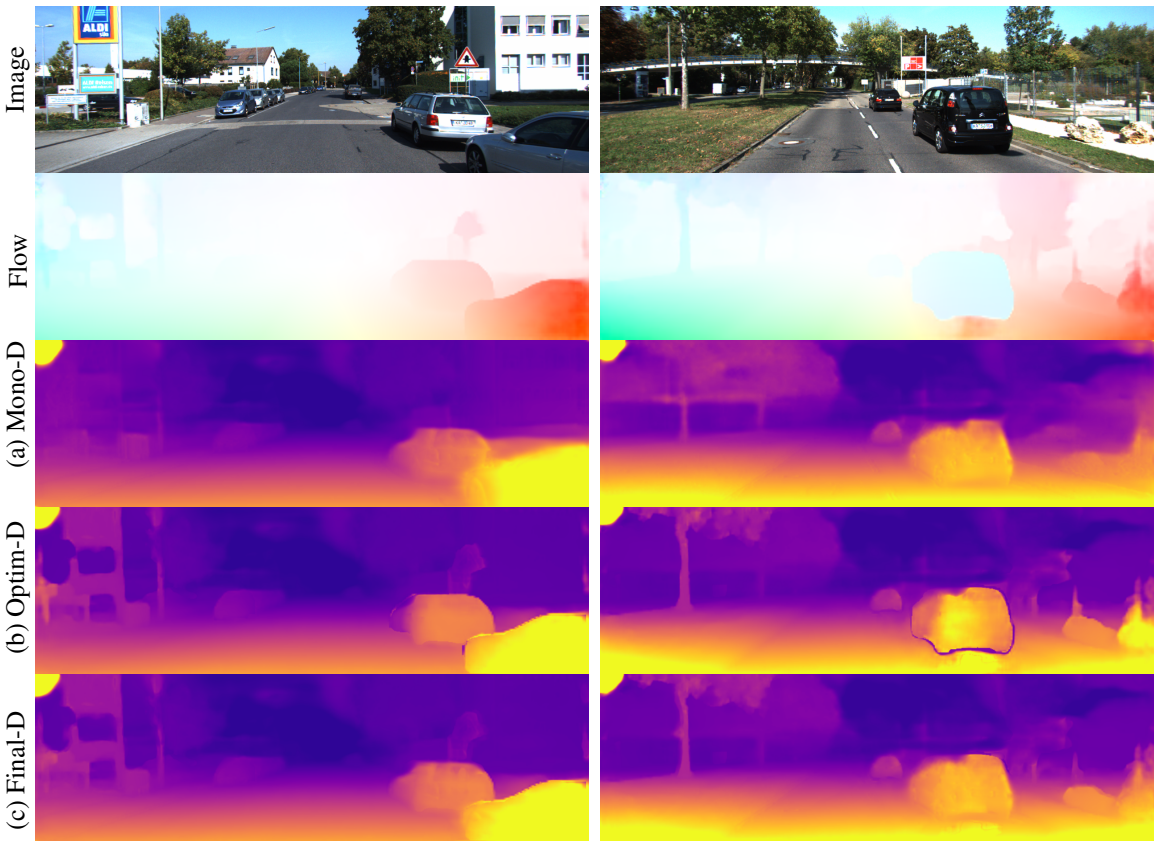


Figure 7.4: **Qualitative visualization of results on KITTI raw test split.** We visualize the estimated optical flow, our initial monocular depth estimation (a), the optimized depth (b) and the final refined depth (c). Our results show that using two-view optical flow as optimization can recover more accurate geometry structure. There are some small issues in some of the instance boundary after optimization, which can be fixed by the network refinement.

- **Optimized-D:** The monocular depth optimized by iteratively Gauss-Newton method, which is the result after the optimization stage in Section 7.3.2.
- **Reweighted-D**: The monocular depth after depth reweighting using output from the Depth Reweighted Network. It is the final result of our method.

7.5.2 Results

Monocular depth estimation on KITTI [167] raw dataset: We evaluate our results following the conventional depth estimation split as Eigen et al. [151]. Our quantita-

Table 7.1: **Quantitative evaluation about monocular depth estimation on KITTI dataset** All method use same the test split as Eigen et al. [151]. Baselines that use multiframe as input are highlighted with †. Baselines that tackle dynamic scenes as part of the objective are higlighted with ‡. Our final result **Reweighted-D** outperform all baseline methods and ablations in both self-supervised and semi-supervised learning setting.

	rmse ↓	rmse_log ↓	abs_rel ↓	seq_rel ↓	$\delta < 1.25 \uparrow$	$\delta < 1.25^2 \uparrow$	$\delta < 1.25^3 \uparrow$
SFMLearner [57] †	6.856	0.283	0.208	1.768	0.678	0.885	0.957
Vid2Depth [157]	6.220	0.250	0.163	1.240	0.762	0.916	0.968
GeoNet [60] ‡	6.090	0.247	0.164	1.303	0.765	0.919	0.968
Godard et al. [56]	5.927	0.247	0.148	1.344	0.803	0.922	0.964
DF-Net [58] ‡	5.507	0.223	0.150	1.124	0.806	0.933	0.973
LK-Learner [75]	5.583	0.228	0.151	1.257	0.810	0.936	0.974
Ranjan et al. [59] †, ‡	5.326	0.217	0.140	1.070	0.826	0.941	0.975
Using self-supervised depth							
Monocular-D	5.301	0.219	0.117	1.057	0.840	0.933	0.969
Optimized-D	**5.091**	0.246	0.138	1.040	0.833	0.937	0.970
Reweighted-D	5.121	**0.211**	**0.116**	**0.975**	**0.853**	**0.942**	0.973
Using semi-supervised depth							
Monocular-D	4.890	0.208	0.134	0.997	0.825	0.942	0.978
Optimized-D	4.586	0.231	0.114	0.891	0.866	0.946	0.975
Reweighted-D	**4.514**	**0.178**	**0.103**	**0.799**	**0.883**	**0.958**	**0.983**
Using DORN[152] depth							
Monocular-D [152]	3.721	0.170	0.112	0.643	0.891	0.962	0.983
Optimized-D	3.759	0.230	0.119	0.755	0.886	0.952	0.975
Reweighted-D	**3.494**	**0.153**	**0.091**	**0.538**	**0.915**	**0.969**	**0.986**

tive comparison to learning monocular depth estimation methods is in Table 7.1 and to classical dynamic reconstruction methods is in Table 7.2. Our final results outperform all baselines and state-of-the-art methods trained semi-supervised as well as self-supervised. Using semi-supervised learning can achieve the best performance in both scale-dependent metrics and scale-invariant metrics. Fig. 7.4 shows additional qualitative examples.

Generalization to unseen videos: We apply the trained model on KITTI datasets directly to the Cityscapes raw videos [168]. Fig. 7.5 visualizes two example frames of our results. Despite the domain differences, our method can better generalize to unseen scenarios and estimate the fine details more accurately.

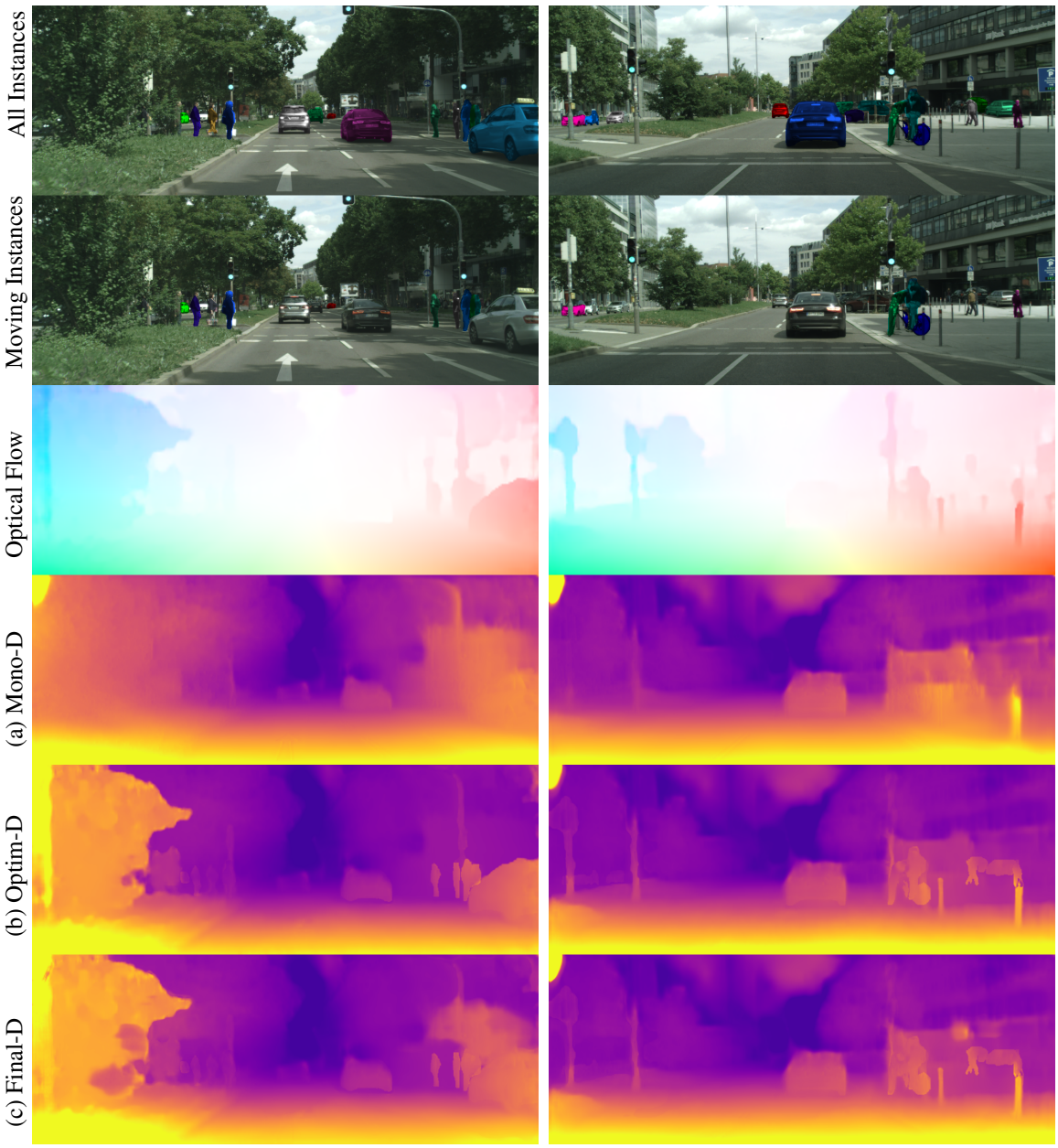


Figure 7.5: **Qualitative visualization of results on Cityscape [168].** We visualized the all the detected instances, all the moving instances and the estimated optical flow. We compare the results of our initial monocular depth estimation (a), the optimized depth (b) and the final refined depth (c). All results are generated by network trained using KITTI data. Our final results (c) show that our method can generalize to unseen videos well.

Table 7.2: **Quantitative comparison to existing monocular dynamic reconstruction methods on KITTI dataset.** Our approach in both self-supervised and semi-supervised setting outperform the existing state-of-the-art dynamic reconstruction methods.

	Use Learning	Number of Views	abs_rel (MRE) ↓
Fragkiadaki et al. [165]	No	Multi-view	0.411
Akhter et al. [166]	No	Multi-view	0.409
Dai et al. [66]	No	Multi-view	0.390
Karsch et al. [154]	Yes	One-view	0.270
Russell et al. [155]	No	Two-view	0.196
Ranftl et al. [68]	No	Two-view	0.148
Kumar et al. [67]	No	Two-view	0.127
Ours self-supervised	Yes	Two-view	0.117
Ours semi-supervised	Yes	Two-view	**0.103**

7.6 Conclusion

We have presented a two-view monocular scene flow method by learning the optimized depth using a single view monocular depth, optical flow, and instance segmentation. Our experiments on real-world KITTI dataset and Cityscapes dataset demonstrate our method can output more accurate depth estimation for geometric structures by leveraging motion-parallax from a pair of monocular views.

We observe the success of the depth estimation relies on the quality of optical flow. If the learned optical flow fails to generalize to particular scenarios, the optimized and refined depth may get affected significantly. If the initial monocular depth estimation deviates too much from the ground truth depth, the optimization may also fail to correct its residual. The recall of the moving instance depends on the instance detection. One direction to improve the current method is to push forward the generalization ability of monocular depth, optical flow, and instance segmentation. Future work can also focus on extending the current approach to multi-frame, which can enable a more robustly optimization to handle outliers across pairs of views.

CHAPTER 8
CONCLUSION AND OUTLOOK

There is rapid progress in computer vision in recent years. Our work in Chapter 4 [90] published in 2016 was the state-of-art method in the KITTI scene flow benchmark [13], taking 80 seconds to run per-frame. Now in July 2019, it is the 18th method in the metric of overall accuracy on the same benchmark. Three methods with better accuracy on the leaderboard can run per-frame within one second, two orders of magnitude faster than our first approach. Such improvements are happening even more rapidly on some of the sub-tasks, such as optical flow on SINTEL [96] and KITTI.

The major takeaway of fast progress is identifying the importance of learning in the task. Deep learning using a massive amount of training data simplifies the traditional non-linear optimization problem. Learning-base algorithms show the potential to effectively regularize the dense output, particularly in the textureless or occluded regions, observed from moving cameras with significant blur, sensor noise, and rapid motion. The forward network inference is usually fast. All of these benefits indicate incorporating learning is a promising direction to achieve fast, dense, and accurate scene flow.

However, vanilla image-based regression approaches suffer from its generalization ability to the unseen data. Recently there is a trend to incorporate traditional paradigm into learning can build the network more data-efficiently, smaller, achieving better accuracy, and generalizing better. It includes using cost-volume and pyramid matching in optical flow [48] and stereo [108], using feedback [55, 136] and unrolled optimization [32, 143].

In this thesis, we target several different representations of scene flow and provide several approaches to synthesize learning and optimization to solve scene flow problem. We believe it can provide valuable insights and baselines for future work. In this chapter, we will conclude our thesis with a summary of our contributions, discussing our limitations,

and an outlook on the several essential problems for future work.

8.1 Summary of Contributions

Efficient Continuous Least-square Optimization of Planar Scene Flow: We represent the dynamic 3D scene as a collection of rigidly moving planar segments and solve it purely as least-square optimization in the continuous domain. With a high-quality correspondence, we developed a novel initialization method to solve our nonlinear scene flow objective using the least-square method progressively in an efficient way. The approach can also potentially benefit from the current learning-based to provide faster, more accurate, dense correspondences as initialization.

Learning Point-based Scene Flow: We propose to use *rigidity* as the representation to disambiguate camera motion and scene motion. With the learned rigidity representation, we can estimate camera motion using least-square as in the rigid region and projected scene flow using computed 2D optical flow. We provide a new semi-synthetic dynamic scene dataset for training and testing the rigidity network. We also include a new guideline for dynamic scene evaluation regarding a large amount of dynamic motion in the scene.

Learning Instance Scene Flow with Unrolled Optimization: We provide a modern synthesis of the classic inverse compositional algorithm for dense image alignment, which can be used to solve rigid instance scene flow. Specifically, we unroll a robust version of the inverse compositional algorithm as several learnable models, which can be trained end-to-end using data. Our method can achieve better accuracy than traditional image alignment methods or image-based regression method and cope well with challenging scenarios with noisy measurements and heavy occlusions.

Learning to Recover Monocular Instance Scene Flow: We propose the first learnable approach that can simultaneously recover depth, 6DoF camera motion, and instance motion in a dynamic scenario. Our method leverages motion-parallax, via jointly estimating

monocular scene flow using an integration of least-square optimization and learning, which ensure the motion and geometry consistency across views. Our evaluation shows the proposed method can benefit from the progress in monocular depth and flow estimation while outperforming them with the synthesis of learning and optimization.

Publications: This cumulative thesis comprises three full-length first-author publications [90, 146, 169] published in top-tier computer vision conferences and one work ready for submission. All of the publications are joint work with the co-authors, listed as the following.

- **Zhaoyang Lv**, Chris Beall, Pablo F Alcantarilla, Fuxin Li, Zsolt Kira, Frank Dellaert, A Continuous Optimization Approach for Efficient and Accurate Scene Flow, *European Conference on Computer Vision*, 2016
- **Zhaoyang Lv**, Kihwan Kim, Alejandro Troccoli, Deqing Sun, James M Rehg, Jan Kautz, Learning Rigidity in Dynamic Scenes with a Moving Camera for 3D Motion Field Estimation, *European Conference on Computer Vision*, 2018
- **Zhaoyang Lv**, Frank Dellaert, James M Rehg, Andreas Geiger, Taking a Deeper Look at the Inverse Compositional Algorithm, *IEEE Conference on Computer Vision and Pattern Recognition*, 2019
- **Zhaoyang Lv**, Kihwan Kim, Alejandro Troccoli, Deqing Sun, James M Rehg, Jan Kautz, Learning to Recover Monocular Depth and Scene Flow from Rigid Moving Scenes, *Ready for Submission*

Open-sourced Softwares and Dataset: This thesis also contributes to several open source softwares and dataset. Click the bold item for the hyperlink.

- **Learning rigidity**: The source code for Chapter 5.
- **Learning Inverse Composition**: . The source code for Chapter 6.

- **RefRESH toolkit**: We use this Blender tool to create the RefRESH dataset in Chapter 5 and the MovingObject3D in Chapter 6. The RefRESH dataset is also released under the same link.

- **GTSAM and Mini-SAM**: Both are least-square optimization libraries. GTSAM was actively developed as the backend optimization for Chapter 4. Mini-SAM achieves similarly functionalities as GTSAM with more flexible support for both CPU and GPU optimization.

8.2 Experimental Insights

The combination of learning and optimization: We find deep learning and optimization are both effective tools in studying scene flow. On the one hand, deep learning can efficiently cope with ill-posed regularization in a complex problem using a massive amount of data. On the other hand, when a well-conditioned least-square problem is identified, solve it using the second-order method such as Gauss-Newton is accurate and fast. This thesis has explored several directions to synthesize optimization and learning. In Chapter 5, we learn scene flow as a dense nonrigid correspondence problem, and we use the rigidity mask as the representation to segment the region for the rigid camera motion. Solving camera motion as least-square optimization in the rigid region is well conditioned, fast, and generate accurate results. In Chapter 6 and Chapter 7, we leverage the instance segmentation to help us recognizing and locating rigid instances. Given the rigid region, we tackle them as a joint optimization of depth and motion.

Integrating learning into optimization: Solving the nonlinear least-square problem also require regularization extensively. The gradient-based optimization problem can be unrolled as a differentiable computational graph, and we can replace the hand-crafted regularization with learnable parameters. We provide a more in-depth look at the photometric alignment in Chapter 6 and geometric alignment in Chapter 7, which can potentially ad-

dress a broader family of optimization problems.

Leveraging top-down and bottom-up segmentation: This thesis extensively leverages grouping and segmentation as middle-level representation. Improving the quality of segmentation using either top-down or bottom-up information are long-standing tasks in computer vision. Using appropriate segmentation can significantly simplify the complexity in the problem and helps us to identify the regions that can be solved via least-square optimization. The success of our learning-based algorithms highly depends on robustly identifying the rigid regions through segmentation. It applies to both the methods in Chapter 5 using bottom-up rigid rigidity segmentation and in Chapter 7 using instance segmentation. Failing to recall the nonrigid region or the moving instances will significantly impact identifying the moving regions, which has been the biggest limitation of the work in this thesis. Meanwhile, there are extensive approaches that focus on improving the diversity of segmentation [170] as well as quality. It will be worthwhile to keep exploring the suitable segmentation representation for the scene flow task.

8.3 Future work

In this thesis, we presented the preliminary explorations into efficient dense scene flow using different representations with a synthesis of learning and optimization. Towards faster and more accurate dense scene flow, future work can exploit multi-frame information, scene understanding, weakly supervised labels, and more expressive subspace representation.

Multi-frames scene flow: This thesis primarily focuses on solving scene flow from two perspective views. Much of the efforts in two-view scene flow has to deal with the natural uncertainty in perspective depth, noisy measurement per frame, and the occlusion problem. Although solving these challenges are essential for dense scene flow representation, the algorithm can be potentially more efficient to address them by incorporating more than two-frame information. Exploiting multi-frame information has been proved to be help-

ful in scene flow [16, 28] as well as optical flow [163, 171, 164]. Potential directions include: (1) exploring frame-to-model scene flow estimation which is closely relevant to dynamic reconstruction [4, 6]; (2) extending Chapter 7 to multi-frame using window-based bundle-adjustment; (3) unroll multi-frame bundle-adjustment as extension to Chapter 6. A fundamental challenge to the multi-frame scene flow is the trade-off between inference speed and accuracy, where a deeper understanding of integrating learning and optimization will play an important role.

Close the loop of scene understanding and scene flow: Semantic labels can provide the top-down regularization on the scene. It can separate the static scene from potential moving scenes. It can also categorize the moving objects into a complete rigid object, articulated object, or fully deformable objects. Semantic or instance ROI is also useful information to achieve foveation in human perception, which may enable lower latency interaction. In Chapter 7, we have shown one approach to recover depth and motion using instance segmentation accurately. Faster and more accurate scene flow can also provide more reliable cross-frame data association for scene understanding. One challenge in closed-loop integration of semantic knowledge and scene flow is the data. While increasing spatial labels are available [172, 173], most of the existing data are for the purely static scenes and particular class, e.g. humans and vehicles. Establishing dynamic scene datasets and exploring the integration of the two tasks will be an essential direction to go.

Towards weakly-supervised learning: Creating large-scale scene flow dataset with supervision requires efforts and new thinking to be scalable. In Chapter 5, we explored REFRESH as one approach, which can be enhanced using 3D reconstruction with semantic information with larger quantities [172] or photo-realistic quality [174]. The recent work in knowledge distillation shows remarkable performance to handle unlabeled videos, as shown in the task of optical flow [164]. Future work can similarly explore distilling the knowledge of the current supervised, trained model to dynamic scenes in the wild.

Exploiting the subspace of Dense 3D motions: In this thesis, we have studied scene flow as planar motion in Chapter 4 and rigid instance motion in Chapter 6 and Chapter 7. All of these approaches cannot apply to the general video scenes as Chapter 5 in which we represent the dynamic motion simply as dense 3D motion vectors. A generalizable underlying representation has not been well exploited. The importance of understanding the subspace of geometry and motion has been widely acknowledged in optical flow[175], depth estimation [160] and human modeling [72]. Learning and optimization the subspace of dense scene flow may bring the opportunity for faster and more accurate solutions.

REFERENCES

[1] S. Vedula et al. “Three-dimensional scene flow”. In: *The IEEE International Conference on Computer Vision (ICCV)*. Vol. 2. 1999, 722–729 vol.2.

[2] C. H. Hung, L. Xu, and J. Y. Jia. “Consistent Binocular Depth and Scene Flow with Chained Temporal Profiles”. In: *International Journal of Computer Vision* 102.1-3 (2013), pp. 271–292.

[3] Pichao Wang et al. “Scene Flow to Action Map: A New Representation for RGB-D Based Action Recognition With Convolutional Neural Networks”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[4] Matthias Innmann et al. “VolumeDeform: Real-time Volumetric Non-rigid Reconstruction”. In: *European Conference on Computer Vision (ECCV)*. 2016.

[5] Richard A Newcombe and Andrew J Davison. “Live dense reconstruction with a single moving camera”. In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE. 2010, pp. 1498–1505.

[6] Richard A. Newcombe, Dieter Fox, and Steven M. Seitz. “DynamicFusion: Reconstruction and Tracking of Non-Rigid Scenes in Real-Time”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[7] Andreas Geiger et al. “3D Traffic Scene Understanding from Movable Platforms”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2014).

[8] Timo Scharwächter et al. “Stixmantics: A Medium-Level Model for Real-Time Semantic Scene Understanding”. In: *European Conference on Computer Vision (ECCV)*. 2014.

[9] A. Shashua, Y. Gdalyahu, and G. Hayun. “Pedestrian Detection for Driving Assistance Systems: Single-frame Classification and System Level Performance”. In: *IEEE Intelligent Vehicles Symposium (IV)*. 2004, pp. 1–6.

[10] A. Wedel et al. “Stereoscopic Scene Flow Computation for 3D Motion Understanding”. In: *International Journal of Computer Vision* 95.1 (2011), pp. 29–51.

[11] Arunkumar Byravan and Dieter Fox. “SE3-Nets: Learning Rigid Body Motion using Deep Neural Networks”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 173–180.

[12] Aseem Behl et al. “Bounding Boxes, Segmentations and Object Coordinates: How Important is Recognition for 3D Scene Flow Estimation in Autonomous Driving Scenarios?” In: *The IEEE International Conference on Computer Vision (ICCV)*. 2017.

[13] M. Menze and A. Geiger. “Object scene flow for autonomous vehicles”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3061–3070.

[14] Zhile Ren et al. “Cascaded Scene Flow Prediction using Semantic Segmentation”. In: *Interntional Conference on 3D Vision*. 2017.

[15] C. Vogel, K. Schindler, and S. Roth. “Piecewise Rigid Scene Flow”. In: *The IEEE International Conference on Computer Vision (ICCV)*. 2013, pp. 1377–1384. ISBN: 1550-5499.

[16] Christoph Vogel, Konrad Schindler, and Stefan Roth. “View-consistent 3d scene flow estimation over multiple frames”. In: *European Conference on Computer Vision (ECCV)*. Vol. 8692. 2014, pp. 263–278. ISBN: 03029743.

[17] F. Huguet and F. Devernay. “A Variational Method for Scene Flow Estimation from Stereo Sequences”. In: *The IEEE International Conference on Computer Vision (ICCV)*. 2007, pp. 1–7. ISBN: 1550-5499.

[18] Philippe Weinzaepfel et al. “DeepFlow: Large Displacement Optical Flow with Deep Matching”. In: *The IEEE International Conference on Computer Vision (ICCV)*. 2013.

[19] A. Dosovitskiy et al. “FlowNet: Learning Optical Flow with Convolutional Networks”. In: *The IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 2758–2766.

[20] Kaiming He et al. “Mask R-CNN”. In: *The IEEE International Conference on Computer Vision (ICCV)*. 2017.

[21] S. Ullman. *The Interpretation of Structure from Motion*. Vol. 203. 1153. The Royal Society, 1979.

[22] João Costeira and Takeo Kanade. “A Multi-body Factorization Method for Independently Moving Objects”. In: *International Journal of Computer Vision* (1997).

[23] Zhengyou Zhang and Olivier D. Faugeras. *3D Dynamic Scene Analysis*. Vol. 27. Springer-Verlag Berlin Heidelberg, 1992.

[24] Allen M Waxman and James H Duncan. “Binocular image flows: Steps toward stereo-motion fusion”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 6 (1986), pp. 715–729.

[25] Zhengyou Zhang and Olivier D. Faugeras. “Estimation of Displacements from Two 3-D Frames Obtained From Stereo”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*. 1992.

[26] Levi Valgaerts et al. “Joint estimation of motion, structure and geometry from stereo sequences”. In: *European Conference on Computer Vision (ECCV)*. Vol. 6314 LNCS. 2010, pp. 568–581. ISBN: 03029743.

[27] T. Basha, Y. Moses, and N. Kiryati. “Multi-view scene flow estimation: A view centered variational approach”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2010, pp. 1506–1513. ISBN: 1063-6919.

[28] Tatsunori Taniai, Sudipta Sinha, and Yoichi Sato. “Fast Multi-frame Stereo Scene Flow with Motion Segmentation”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

[29] Eddy Ilg et al. “Occlusions, Motion and Depth Boundaries with a Generic Network for Disparity, Optical Flow or Scene Flow Estimation”. In: *European Conference on Computer Vision (ECCV)*. 2018.

[30] Huaizu Jiang et al. “SENSE: A Shared Encoder Network for Scene-flow Estimation”. In: *The IEEE International Conference on Computer Vision (ICCV)*. 2019.

[31] Nikolaus Mayer et al. “A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 4040–4048.

[32] Wei-Chiu Ma et al. “Deep Rigid Instance Scene Flow”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[33] E. Herbst, X. Ren, and D. Fox. “RGB-D flow: Dense 3-D motion estimation using color and depth”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2013, pp. 2276–2282. ISBN: 1050-4729.

[34] Antoine Letouzey, Benjamin Petit, and Edmond Boyer. “Scene Flow from Depth and Color Images”. In: *British Machine Vision Conference (BMVC)*. BMVA Press, 2011, 46:1–11.

[35] M. Jaimez et al. “A primal-dual framework for real-time dense RGB-D scene flow”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 98–104. ISBN: 1050-4729.

[36] Deqing Sun, Erik B Sudderth, and Hanspeter Pfister. “Layered RGBD scene flow estimation”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2015, pp. 548–556.

[37] M. Jaimez et al. “Motion Cooperation: Smooth Piece-wise Rigid Scene Flow from RGB-D Images”. In: *Interntional Conference on 3D Vision*. 2015, pp. 64–72.

[38] Julian Quiroga et al. “Dense semi-rigid scene flow estimation from RGB-D images”. In: *European Conference on Computer Vision (ECCV)*. Vol. 8695 LNCS. 2014, pp. 567–582. ISBN: 03029743.

[39] M. Jaimez et al. “Fast Odometry and Scene Flow from RGB-D Cameras based on Geometric Clustering”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2017.

[40] Arash K Ushani et al. “A learning approach for real-time temporal scene flow estimation from LIDAR data”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017.

[41] Aseem Behl et al. “PointFlowNet: Learning Representations for Rigid Motion Estimation from Point Clouds”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[42] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. “Flownet3d: Learning scene flow in 3d point clouds”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 529–537.

[43] Minglun Gong. “Real-time joint disparity and disparity flow estimation on programmable graphics hardware”. In: *Computer Vision and Image Understanding* 113.1 (2009), pp. 90–100.

[44] Andreas Wedel et al. “Efficient dense scene flow from sparse or dense stereo data”. In: *European Conference on Computer Vision (ECCV)*. Vol. 5302 LNCS. 2008, pp. 739–751. ISBN: 03029743.

[45] Clemens Rabe et al. “Dense, robust, and accurate motion field estimation from stereo image sequences in real-time”. In: *European Conference on Computer Vision (ECCV)*. Springer-Verlag, 2010, pp. 582–595.

[46] E. Ilg et al. “FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[47] Anurag Ranjan and Michael J Black. “Optical flow estimation using a spatial pyramid network”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[48] Deqing Sun et al. “PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[49] Alex Kendall, Matthew Grimes, and Roberto Cipolla. “PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization”. In: *The IEEE International Conference on Computer Vision (ICCV)*. 2015.

[50] Alex Kendall and Roberto Cipolla. “Geometric loss functions for camera pose regression with deep learning”. In: *The IEEE International Conference on Computer Vision (ICCV)*. 2017.

[51] Samarth Brahmbhatt et al. “Geometry-Aware Learning of Maps for Camera Localization”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[52] Fabian Manhardt et al. “Deep Model-Based 6D Pose Refinement in RGB”. In: *European Conference on Computer Vision (ECCV)*. 2018.

[53] Martin Sundermeyer et al. “Implicit 3D Orientation Learning for 6D Object Detection from RGB Images”. In: *European Conference on Computer Vision (ECCV)*. 2018.

[54] B. Ummenhofer et al. “DeMoN: Depth and Motion Network for Learning Monocular Stereo”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[55] H. Zhou, B. Ummenhofer, and T. Brox. “DeepTAM: Deep Tracking and Mapping”. In: *European Conference on Computer Vision (ECCV)*. 2018.

[56] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. “Unsupervised monocular depth estimation with left-right consistency”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 270–279.

[57] Tinghui Zhou et al. “Unsupervised Learning of Depth and Ego-Motion from Video”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[58] Yuliang Zou, Zelun Luo, and Jia-Bin Huang. “Df-net: Unsupervised joint learning of depth and flow using cross-task consistency”. In: *European Conference on Computer Vision (ECCV)*. 2018, pp. 36–53.

[59] Anurag Ranjan et al. “Competitive Collaboration: Joint Unsupervised Learning of Depth, Camera Motion, Optical Flow and Motion Segmentation”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[60] Zhichao Yin and Jianping Shi. “GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[61] Zhe Cao et al. “Learning Independent Object Motion from Unlabelled Stereoscopic Videos”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[62] Carlo Tomasi and Takeo Kanade. “Shape and motion from image streams under orthography: a factorization method”. In: *International Journal of Computer Vision* 9.2 (1992), pp. 137–154.

[63] Christoph Bregler. “Recovering non-rigid 3D shape from image streams”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2000.

[64] Bregler Chris, Torresani Lorenzo, and Hertzmann Aaron. “Nonrigid Structure-from-Motion: Estimating Shape and Motion with Hierarchical Priors”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 30 (2007).

[65] Ijaz Akhter et al. “Nonrigid Structure from Motion in Trajectory Space”. In: *Advances in Neural Information Processing Systems (NIPS)*. Curran Associates, Inc., 2009, pp. 41–48.

[66] Yuchao Dai, Hongdong Li, and Mingyi He. “A simple prior-free method for non-rigid structure-from-motion factorization”. In: *International Journal of Computer Vision* 107.2 (2014), pp. 101–122.

[67] Suryansh Kumar, Yuchao Dai, and Hongdong Li. “Monocular dense 3d reconstruction of a complex dynamic scene from two perspective frames”. In: *The IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 4649–4657.

[68] Rene Ranftl et al. “Dense monocular depth estimation in complex dynamic scenes”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 4058–4066.

[69] Richard A. Newcombe et al. “KinectFusion: Real-time Dense Surface Mapping and Tracking”. In: *IEEE and ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR)*. Washington, DC, USA: IEEE Computer Society, 2011, pp. 127–136. ISBN: 978-1-4577-2183-0.

[70] Mingsong Dou et al. “Fusion4D: Real-time Performance Capture of Challenging Scenes”. In: *SIGGRAPH*. 2016.

[71] Tao Yu et al. “Doublefusion: Real-time capture of human performances with inner body shapes from a single depth sensor”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 7287–7296.

[72] Matthew Loper et al. “SMPL: A Skinned Multi-Person Linear Model”. In: *SIGGRAPH Asia* 34.6 (Oct. 2015), 248:1–248:16.

[73] T D Barfoot. *State Estimation for Robotics*. Cambridge University Press, 2017. ISBN: 9781107159396.

[74] Jose-Luis Blanco. *A tutorial on se (3) transformation parameterizations and on-manifold optimization*. Tech. rep.

[75] Chaoyang Wang et al. “Learning Depth From Monocular Videos Using Direct Methods”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[76] Frank Dellaert and Michael Kaess. “Factor Graphs for Robot Perception”. In: *Foundations and Trends® in Robotics* 6.1-2 (2017), pp. 1–139.

[77] Frank Dellaert and Robert Collins. “Fast image-based tracking by selective pixel integration”. In: *The IEEE International Conference on Computer Vision (ICCV) Workshops*. 1999.

[78] Zhengyou Zhang, Programme Robotique, and Projet Robotvis. “Parameter estimation techniques: a tutorial with application to conic fitting”. In: *Image and Vision Computing* 15 (1997), pp. 59–76.

[79] Jonathan T. Barron. “A General and Adaptive Robust Loss Function”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[80] Frank Dellaert and Michael Kaess. “Square Root SAM: Simultaneous localization and mapping via square root information smoothing”. In: *International Journal of Robotics Research* 25.12 (2006), pp. 1181–1203.

[81] Christoph Vogel, Stefan Roth, and Konrad Schindler. “An Evaluation of Data Costs for Optical Flow”. In: *German Conference on Pattern Recognition (GCPR)*. 2013.

[82] C. Lawrence Zitnick and Piotr Dollár. “Edge Boxes: Locating Object Proposals from Edges”. In: *European Conference on Computer Vision (ECCV)*. 2014.

[83] Koichiro Yamaguchi, David McAllester, and Raquel Urtasun. “Efficient Joint Segmentation, Occlusion Labeling, Stereo and Flow Estimation”. In: *European Conference on Computer Vision (ECCV)*. 2014, pp. 756–771. ISBN: 978-3-319-10602-1.

[84] C. Vogel, K. Schindler, and S. Roth. “3D Scene Flow Estimation with a Piecewise Rigid Scene Model”. In: *International Journal of Computer Vision* 115.1 (2015), pp. 1–28.

[85] Michael Hornacek, Andrew Fitzgibbon, and Carsten Rother. “SphereFlow: 6 DoF Scene Flow from RGB-D Pairs”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014.

[86] Laura Sevilla-Lara et al. “Optical Flow with Semantic Segmentation and Localized Layers”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[87] Moritz Menze, Christian Heipke, and Andreas Geiger. “Discrete Optimization for Optical Flow”. In: *German Conference on Pattern Recognition (GCPR)*. 2015.

[88] Jerome Revaud et al. “EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[89] Jonas Wulff, Laura Sevilla-Lara, and Michael J. Black. “Optical Flow in Mostly Rigid Scenes”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[90] Zhaoyang Lv et al. “A Continuous Optimization Approach for Efficient and Accurate Scene Flow”. In: *European Conference on Computer Vision (ECCV)*. Springer. 2016, pp. 757–773.

[91] Gül Varol et al. “Learning from Synthetic Humans”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[92] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2nd ed. New York, NY, USA: Cambridge University Press, 2003. ISBN: 0521540518.

[93] David Nistér. “Preemptive RANSAC for Live Structure and Motion Estimation”. In: *The IEEE International Conference on Computer Vision (ICCV)*. 2003.

[94] Sudheendra Vijayanarasimhan et al. “SfM-Net: Learning of Structure and Motion from Video”. In: *arXiv* abs/1704.07804 (2017).

[95] Angela Dai et al. “BundleFusion: Real-time Globally Consistent 3D Reconstruction using On-the-fly Surface Re-integration”. In: *ACM Transactions on Graphics 2017 (TOG)* (2017).

[96] Daniel J. Butler et al. “A Naturalistic Open Source Movie for Optical Flow Evaluation”. In: *European Conference on Computer Vision (ECCV)*. The Royal Society, 2012, pp. 611–625. ISBN: 978-3-642-33783-3.

[97] Stephan R. Richter, Zeeshan Hayder, and Vladlen Koltun. “Playing for Benchmarks”. In: *The IEEE International Conference on Computer Vision (ICCV)*. 2017.

[98] Hao Su et al. “Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views”. In: *The IEEE International Conference on Computer Vision (ICCV)*. 2015.

[99] A. Handa et al. “A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China, 2014.

[100] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. “SUN3D: A Database of Big Spaces Reconstructed Using SfM and Object Labels”. In: *The IEEE International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, 2013, pp. 1625–1632.

[101] Catalin Ionescu et al. “Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 36.7 (2014), pp. 1325–1339.

[102] Gerard Pons-Moll et al. “Dyna: A Model of Dynamic Human Shape in Motion”. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 34.4 (Aug. 2015), 120:1–120:14.

[103] Kaiming He et al. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”. In: *The IEEE International Conference on Computer Vision (ICCV)*. Washington, DC, USA: IEEE Computer Society, 2015, pp. 1026–1034. ISBN: 978-1-4673-8391-2.

[104] J. Sturm et al. “A Benchmark for the Evaluation of RGB-D SLAM Systems”. In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2012.

[105] Raúl Mur-Artal and Juan D. Tardós. “ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras”. In: *IEEE Transactions on Robotics* 33.5 (2017), pp. 1255–1262.

[106] Liang-Chieh Chen et al. “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2017).

[107] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. “Fast Global Registration”. In: *European Conference on Computer Vision (ECCV)*. 2016, pp. 766–782.

[108] Jia-Ren Chang and Yong-Sheng Chen. “Pyramid Stereo Matching Network”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 5410–5418.

[109] Bruce D Lucas and Takeo Kanade. “An iterative image registration technique with an application to stereo vision”. In: *Intl. Joint Conf. on AI (IJCAI)*. Vol. 81. 1981, pp. 674–679.

[110] Daniel Scharstein and Richard Szeliski. “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms”. In: *International Journal of Computer Vision* 47 (2002), pp. 7–42.

[111] Chaoyang Wang et al. “Deep-LK for Efficient Adaptive Object Tracking”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2018.

[112] Adel Bibi, Matthias Mueller, and Bernard Ghanem. “Target Response Adaptation for Correlation Filter Tracking”. In: *European Conference on Computer Vision (ECCV)*. 2016.

[113] Jianbo Shi and C. Tomasi. “Good features to track”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1994.

[114] Matthew Brown and David Lowe. “Automatic Panoramic Image Stitching using Invariant Features”. In: *International Journal of Computer Vision* 74.1 (2007), pp. 59–73.

[115] Anat Levin et al. “Seamless image stitching in the gradient domain”. In: *European Conference on Computer Vision (ECCV)*. 2004.

[116] Heung-Yeung Shum and Richard Szeliski. “Systems and Experiment Paper: Construction of Panoramic Image Mosaics with Global and Local Alignment”. In: *International Journal of Computer Vision* 36.2 (2000), pp. 101–130.

[117] Michal Irani and Shmuel Peleg. “Improving resolution by image registration”. In: *CVGIP: Graphical models and image processing* 53.3 (1991), pp. 231–239.

[118] Ronald Clark et al. “Learning to Solve Nonlinear Least Squares for Monocular Stereo”. In: *European Conference on Computer Vision (ECCV)*. Sept. 2018.

[119] J. Engel, T. Schöps, and D. Cremers. "LSD-SLAM: Large-Scale Direct Monocular SLAM". In: *European Conference on Computer Vision (ECCV)*. 2014.

[120] Jakob Engel, Vladlen Koltun, and Daniel Cremers. "Direct Sparse Odometry". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 40 (2018), pp. 611–625.

[121] Simon Baker and Iain Matthews. *Lucas-Kanade 20 Years On: A Unifying Framework: Part 1*. Tech. rep. Pittsburgh, PA: Carnegie Mellon University, 2002.

[122] Simon Baker et al. *Lucas-Kanade 20 Years On: A Unifying Framework: Part 2*. Tech. rep. Carnegie Mellon University, 2003.

[123] Peter J. Huber. "Robust Estimation of a Location Parameter". In: *The Annals of Mathematical Statistics* 35.1 (Mar. 1964), pp. 73–101.

[124] Ake Björck. *Numerical Methods for Least Squares Problems*. Siam Philadelphia, 1996.

[125] Donald W. Marquardt. "An Algorithm for Least-Squares Estimation of Nonlinear Parameters". In: *SIAM J. on Algebraic and Discrete Methods* 11.2 (1963), pp. 431–441.

[126] Berthold KP Horn and EJ Weldon. "Direct methods for recovering motion". In: *International Journal of Computer Vision* 2.1 (1988), pp. 51–76.

[127] Michal Irani and P. Anandan. "About Direct Methods". In: *The IEEE International Conference on Computer Vision (ICCV) Workshops*. 2000.

[128] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. "DTAM: Dense tracking and mapping in real-time". In: *The IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2011, pp. 2320–2327.

[129] Richard Szeliski. "Image alignment and stitching: A tutorial". In: *Foundations and Trends in Computer Graphics and Vision* 2.1 (2007), pp. 1–104.

[130] Michael J Black and Paul Anandan. "The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields". In: *Computer Vision and Image Understanding* 63.1 (1996), pp. 75–104.

[131] P. Anandan. "A computational framework and an algorithm for the measurement of visual motion". In: *International Journal of Computer Vision* 2.3 (1989), pp. 283–310.

[132] Frédéric Jurie and Michel Dhome. “Hyperplane approximation for template matching”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 24.7 (2002), pp. 996–1000.

[133] Shih-En Wei et al. “Convolutional pose machines”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 4724–4732.

[134] Angjoo Kanazawa et al. “End-to-end Recovery of Human Shape and Pose”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[135] Chen-Hsuan Lin and Simon Lucey. “Inverse Compositional Spatial Transformer Networks”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[136] Yi Li et al. “DeepIM: Deep Iterative Matching for 6D Pose Estimation”. In: *European Conference on Computer Vision (ECCV)*. 2018.

[137] Ian Cherabier et al. “Learning Priors for Semantic 3D Reconstruction”. In: *European Conference on Computer Vision (ECCV)*. 2018.

[138] Gernot Riegler, Matthias Rüther, and Horst Bischof. “ATGV-Net: Accurate Depth Super-Resolution”. In: *European Conference on Computer Vision (ECCV)*. 2016.

[139] Erich Kobler et al. “Variational Networks: Connecting Variational Methods and Deep Learning”. In: *German Conference on Pattern Recognition (GCPR)*. 2017.

[140] Tim Meinhardt et al. “Learning Proximal Operators: Using Denoising Networks for Regularizing Inverse Imaging Problems”. In: *The IEEE International Conference on Computer Vision (ICCV)*. 2017.

[141] René Ranftl and Thomas Pock. “A Deep Variational Model for Image Segmentation”. In: *German Conference on Pattern Recognition (GCPR)*. 2014.

[142] René Ranftl and Vladlen Koltun. “Deep Fundamental Matrix Estimation”. In: *European Conference on Computer Vision (ECCV)*. Sept. 2018.

[143] Chengzhou Tang and Ping Tan. “BA-Net: Dense Bundle Adjustment Network”. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2019.

[144] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *Proceedings of the International Conference on Learning Representations (ICLR)* (2015).

[145] Angel X. Chang et al. *ShapeNet: An Information-Rich 3D Model Repository*. Tech. rep. arXiv:1512.03012 [cs.GR]. Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[146] Zhaoyang Lv et al. "Learning Rigidity in Dynamic Scenes with a Moving Camera for 3D Motion Field Estimation". In: *European Conference on Computer Vision (ECCV)*. 2018.

[147] Yang Chen and Gérard Medioni. "Object modelling by registration of multiple range images". In: *Image and Vision Computing* 10.3 (1992), pp. 145–155.

[148] Paul J. Besl and Neil D. McKay. "A Method for Registration of 3-D Shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 14.2 (Feb. 1992), pp. 239–256.

[149] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. "Open3D: A Modern Library for 3D Data Processing". In: *arXiv:1801.09847* (2018).

[150] Frank Steinbrücker, Jürgen Sturm, and Daniel Cremers. "Real-time visual odometry from dense RGB-D images". In: *The IEEE International Conference on Computer Vision (ICCV) Workshops*. IEEE. 2011, pp. 719–722.

[151] David Eigen, Christian Puhrsch, and Rob Fergus. "Depth map prediction from a single image using a multi-scale deep network". In: *Advances in Neural Information Processing Systems (NIPS)*. 2014, pp. 2366–2374.

[152] Huan Fu et al. "Deep Ordinal Regression Network for Monocular Depth Estimation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[153] T. Basha, Y. Moses, and N. Kiryati. "Multi-view Scene Flow Estimation: A View Centered Variational Approach". In: *International Journal of Computer Vision* 101.1 (2013), pp. 6–21.

[154] Kevin Karsch, Ce Liu, and Sing Bing Kang. "Depth transfer: Depth extraction from video using non-parametric sampling". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 36.11 (2014), pp. 2144–2158.

[155] Chris Russell, Rui Yu, and Lourdes Agapito. "Video pop-up: Monocular 3d reconstruction of dynamic scenes". In: *European Conference on Computer Vision (ECCV)*. Springer. 2014, pp. 583–598.

[156] Ravi Garg et al. "Unsupervised cnn for single view depth estimation: Geometry to the rescue". In: *European Conference on Computer Vision (ECCV)*. Springer. 2016, pp. 740–756.

[157] Reza Mahjourian, Martin Wicke, and Anelia Angelova. “Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 5667–5675.

[158] Keisuke Tateno et al. “Cnn-slam: Real-time dense monocular slam with learned depth prediction”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6243–6252.

[159] Nan Yang et al. “Deep Virtual Stereo Odometry: Leveraging Deep Depth Prediction for Monocular Direct Sparse Odometry”. In: *European Conference on Computer Vision (ECCV)*. 2018.

[160] Michael Bloesch et al. “CodeSLAM—learning a compact, optimisable representation for dense visual SLAM”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 2560–2568.

[161] Zhengqi Li et al. “Learning the Depths of Moving People by Watching Frozen People”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[162] Yevhen Kuznietsov, Jörg Stückler, and Bastian Leibe. “Semi-Supervised Deep Learning for Monocular Depth Map Prediction”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[163] Joel Janai et al. “Unsupervised Learning of Multi-Frame Optical Flow with Occlusions”. In: *European Conference on Computer Vision (ECCV)*. Vol. Lecture Notes in Computer Science, vol 11220. Springer, Cham, Sept. 2018, pp. 713–731.

[164] Pengpeng Liu et al. “SelFlow: Self-Supervised Learning of Optical Flow”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 4571–4580.

[165] Katerina Fragkiadaki et al. “Grouping-based low-rank trajectory completion and 3D reconstruction”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2014, pp. 55–63.

[166] Ijaz Akhter et al. “Trajectory space: A dual representation for nonrigid structure from motion”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 33.7 (2010), pp. 1442–1456.

[167] A. Geiger, P. Lenz, and R. Urtasun. “Are we ready for autonomous driving? The KITTI vision benchmark suite”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012, pp. 3354–3361. ISBN: 1063-6919.

[168] Marius Cordts et al. “The Cityscapes Dataset for Semantic Urban Scene Understanding”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[169] Zhaoyang Lv et al. “Taking a Deeper Look at the Inverse Compositional Algorithm”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[170] Ronghang Hu et al. “Learning to segment every thing”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 4233–4241.

[171] Daniel Maurer, Michael Stoll, and Andrés Bruhn. “Directional Priors for Multi-Frame Optical Flow.” In: *British Machine Vision Conference (BMVC)*. 2018, p. 106.

[172] Angela Dai et al. “ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[173] Alexander Kirillov et al. “Panoptic segmentation”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 9404–9413.

[174] Julian Straub et al. “The Replica Dataset: A Digital Replica of Indoor Spaces”. In: *arXiv preprint arXiv:1906.05797* (2019).

[175] Jonas Wulff and Michael J. Black. “Efficient Sparse-to-Dense Optical Flow Estimation using a Learned Basis and Layers”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015, pp. 120–130.